

March 2022

DISTRIBUTED LEARNING ALGORITHMS: COMMUNICATION EFFICIENCY AND ERROR RESILIENCE

Raj Kumar Maity
University of Massachusetts Amherst

Follow this and additional works at: https://scholarworks.umass.edu/dissertations_2



Part of the [Computer Engineering Commons](#)

Recommended Citation

Maity, Raj Kumar, "DISTRIBUTED LEARNING ALGORITHMS: COMMUNICATION EFFICIENCY AND ERROR RESILIENCE" (2022). *Doctoral Dissertations*. 2457.
<https://doi.org/10.7275/26438407> https://scholarworks.umass.edu/dissertations_2/2457

This Open Access Dissertation is brought to you for free and open access by the Dissertations and Theses at ScholarWorks@UMass Amherst. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact scholarworks@library.umass.edu.

**DISTRIBUTED LEARNING ALGORITHMS:
COMMUNICATION EFFICIENCY AND ERROR
RESILIENCE**

A Dissertation Presented

by

RAJ KUMAR MAITY

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2022

College of Information and Computer Sciences

© Copyright by Raj Kumar Maity 2022

All Rights Reserved

DISTRIBUTED LEARNING ALGORITHMS: COMMUNICATION EFFICIENCY AND ERROR RESILIENCE

A Dissertation Presented

by

RAJ KUMAR MAITY

Approved as to style and content by:

Arya Mazumdar, Chair

Gerome Miklau, Member

Cameron Musco, Member

Markos A. Katsoulakis, Member

James Allan, Chair of the Faculty
College of Information and Computer Sciences

ACKNOWLEDGMENTS

I would like to acknowledge the people who have played a crucial role in my PhD life. This thesis would not be possible without the help of some amazing people I have met in Umass Amherst.

Fast and foremost, I would like to thank my advisor, Professor Arya Mazumdar for his invaluable guidance over the last five years. Throughout the years, Arya has been a great support and inspiration to me. My research journey began with a course project in *Coding theory and application* taught by Arya. This project paved the pathway for my research in distributed learning. Through many discussions with Arya, I learnt how to understand, solve and explain problems in simple manner. Also, he has given me the freedom to pursue problems and explore different interests throughout my PhD. I will always be indebted to him for his wisdom and directions and above all, his patience and tolerance towards my stupidity.

I would like to also thank my thesis committee members who were very kind to be part of it. I would like to especially thank Professor Gerome Miklau who was also my synthesis project advisor and a big influence behind my interest in differential privacy. Also, I would like to thank Professor Cameron Musco who has been a positive force towards my interest in theoretical computer science and numerical linear algebra and has advised me in several aspects in my research.

I would like to acknowledge the teaching of some excellent teachers in University of Massachusetts, Amherst. My knowledge and ideas in the field of theoretical computer science, optimization, machine learning have been enriched and shaped by the superb teaching of Professor Akhsay Krishnamurthy, Professor Barna Saha, Professor Andrew

Mcgregor and my advisor Prof Arya Mazumdar. I have also been fortunate to work as teaching assistant in several undergraduate and graduate courses. This gave me a unique perspective in learning new topics and improved my presentation skill.

Over the time of graduate studies, I had the privilege to work with some extraordinary talented people on research projects. I had the fortune to work with Ankit Singh Rawat and Venkata Gandikota (GV) while they were post doctoral fellow at Umass Amherst. I was greatly benefited from their research experience. Other than that, I had a very productive and enjoyable collaboration with my friends Ryan McKenna (from Database lab working on differential privacy) and Soumyabrata Pal. I would like to especially acknowledge my friend Avishek Ghosh (UC Berkeley) who has been extremely influential in my research.

The journey of the PhD is a long one. The stay in Amherst has been a very enjoyable one with the time that I get spent with my friends. I really like to thank Ryan, Soumyabrata, Anil, Abhinabh, Nitish, Raghab, Anirudh, Sarwar, Anna and Iro for their support with my studies and every day life. The cheerful interaction with Soumyasundar Pal (McGill University) has been very helpful in reducing the stress of research life.

Finally I like to thank my parents and my sisters. Their support and inspiration has always been a driving force in my life.

I gratefully acknowledge the supports of the NSF grants CCF 1642658, CCF 1618512, CCF 1934846 and 19019046 at different stages of the thesis.

ABSTRACT

DISTRIBUTED LEARNING ALGORITHMS: COMMUNICATION EFFICIENCY AND ERROR RESILIENCE

FEBRUARY 2022

RAJ KUMAR MAITY

B.E., JADAVPUR UNIVERSITY

M.E., INDIAN INSTITUTE OF SCIENCE

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Arya Mazumdar

In modern day machine learning applications such as self-driving cars, recommender systems, robotics, genetics etc., the size of the training data has grown to the point that it has become essential to design distributed learning algorithms. A general framework for the distributed learning is *data parallelism* where the data is distributed among the *worker machines* for parallel processing and computation to speed up learning. With billions of devices such as cellphones, computers etc., the data is inherently distributed and stored locally in the users' devices. Learning in this set up is popularly known as *Federated Learning*. The speed-up due to distributed framework gets hindered by some fundamental problems such as straggler workers, communication bottleneck due to high communication overhead between workers and central server, adversarial failure popularly known as *Byzantine failure*. In this thesis, we study and develop distributed algorithms that are error resilient and communication efficient.

First, we address the problem of straggler workers where the learning is delayed due to slow workers in the distributed setup. To mitigate the effect of the stragglers, we employ **LDPC** (low density parity check) code to encode the data and implement gradient descent algorithm in the distributed setup. Second, we present a family of vector quantization schemes *vqSGD* (vector quantized Stochastic Gradient Descent) that provides an asymptotic reduction in the communication cost with convergence guarantees in the first order distributed optimization. We also showed that *vqSGD* provides strong privacy guarantee. Third, we address the problem of Byzantine failure together with communication-efficiency in the first order gradient descent algorithm. We consider a generic class of δ - approximate compressor for communication efficiency and employ a simple *norm based thresholding* scheme to make the learning algorithm robust to Byzantine failures. We establish statistical error rate for non-convex smooth loss. Moreover, we analyze the compressed gradient descent algorithm with error feedback in a distributed setting and in the presence of Byzantine worker machines. Fourth, we employ the generic class of δ - approximate compressor to develop a communication efficient second order Newton-type algorithm and provide rate of convergence for smooth objective. Fifth, we propose **COMRADE** (COMmunication-efficient and Robust Approximate Distributed nEwton), an iterative second order algorithm that is communication efficient as well as robust against Byzantine failures. Sixth, we propose a distributed *cubic-regularized Newton* algorithm that can escape saddle points effectively for non-convex loss function and find a local minima . Furthermore, the proposed algorithm can resist the attack of the Byzantine machines, which may create *fake local minima* near the saddle points of the loss function, also known as saddle-point attack.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS	iv
ABSTRACT	vi
LIST OF TABLES	xi
LIST OF FIGURES	xii
CHAPTER DEPENDENCIES	xv
NOTATION	xvi
 CHAPTER	
1. INTRODUCTION	1
2. ROBUST GRADIENT DESCENT VIA MOMENT ENCODING AND LDPC CODES	11
2.1 System Model and Background	15
2.1.1 The Data Coding Method of [84] and the Gradient Coding Approach of [120]	18
2.2 Encoding Second Moment : Optimization with Approximate Gradient	21
2.2.1 Exact Computation of Gradient in Each Step	21
2.2.2 Approximate Recovery of Gradient in Every Step	24
2.3 Simulation Results	32
2.4 Conclusion and Future Direction	36

3. VQSGD: VECTOR QUANTIZED STOCHASTIC GRADIENT DESCENT	37
3.1 Related Work	41
3.2 Preliminaries	42
3.3 Quantization Scheme	44
3.4 Constructions of Point Sets and Lower Bound	48
3.4.1 Gaussian Point Set	50
3.4.2 Derandomizing with Reed Muller Codes	53
3.4.3 Other Deterministic Constructions	54
3.4.3.1 Cross Polytope Scheme	55
3.4.3.2 Scaled ε -nets	56
3.5 Private Quantization	57
3.5.1 Randomized Response	64
3.5.2 Privacy Using RAPPOR	67
3.6 Experiments	70
3.7 Conclusion and Future Direction	74
4. COMMUNICATION-EFFICIENT AND BYZANTINE-ROBUST DISTRIBUTED LEARNING WITH ERROR FEEDBACK	75
4.1 Problem Formulation	80
4.2 Compression at Worker Machines	81
4.3 Robust Compressed Gradient Descent	83
4.4 Distributed Learning with Restricted Adversaries	85
4.4.1 Main Results	86
4.5 Distributed Optimization with Arbitrary Adversaries	99
4.5.1 Main Results	100
4.6 Byzantine Robust Distributed Learning with Error Feedback	104
4.6.1 Main Results	106
4.7 Experiments	119
4.8 Conclusion and Future Direction	123
5. COMMUNICATION EFFICIENT DISTRIBUTED APPROXIMATE NEWTON METHOD	125

5.1	Background and Problem Statement	126
5.2	One Round Compression	131
5.3	Two Round Compression	141
5.4	Experimental Result	146
5.5	Conclusion and Future Direction.....	148
6.	DISTRIBUTED NEWTON CAN COMMUNICATE LESS AND RESIST BYZANTINE WORKERS	150
6.1	Problem Formulation	153
6.2	COMRADE Can Communicate Less	154
6.2.1	Theoretical Guarantee	156
6.3	COMRADE Can Resist Byzantine Workers	168
6.4	COMRADE Can Communicate Even Less and Resist Byzantine Workers	172
6.5	Experimental Results	175
6.6	Conclusion and Future Direction.....	178
7.	ESCAPING SADDLE POINTS IN DISTRIBUTED NEWTON'S METHOD WITH BYZANTINE RESILIENCE	179
7.1	Problem Formulation	180
7.1.1	Our Contributions.....	181
7.2	Related Work	182
7.3	Distributed Cubic Regularized Newton	183
7.3.1	Some Useful Facts	184
7.3.2	Theoretical Guarantees	186
7.3.3	Solution of the Cubic Sub-Problem	194
7.4	Byzantine Resilience	197
7.5	Experimental Results	206
7.6	Conclusion and Future Direction.....	209
	BIBLIOGRAPHY	211

LIST OF TABLES

Table		Page
2.1	List of notation	16
3.1	List of results. (N : number of worker nodes, d : dimension).	39
3.2	(Up): Comparison of non private quantization schemes. (Down): Comparison of private quantization schemes. (N : number of worker nodes, s, c : tuning parameter (≥ 1))	40
3.3	Comparison in classification error (mean \pm standard deviation) for various UCI datasets	73

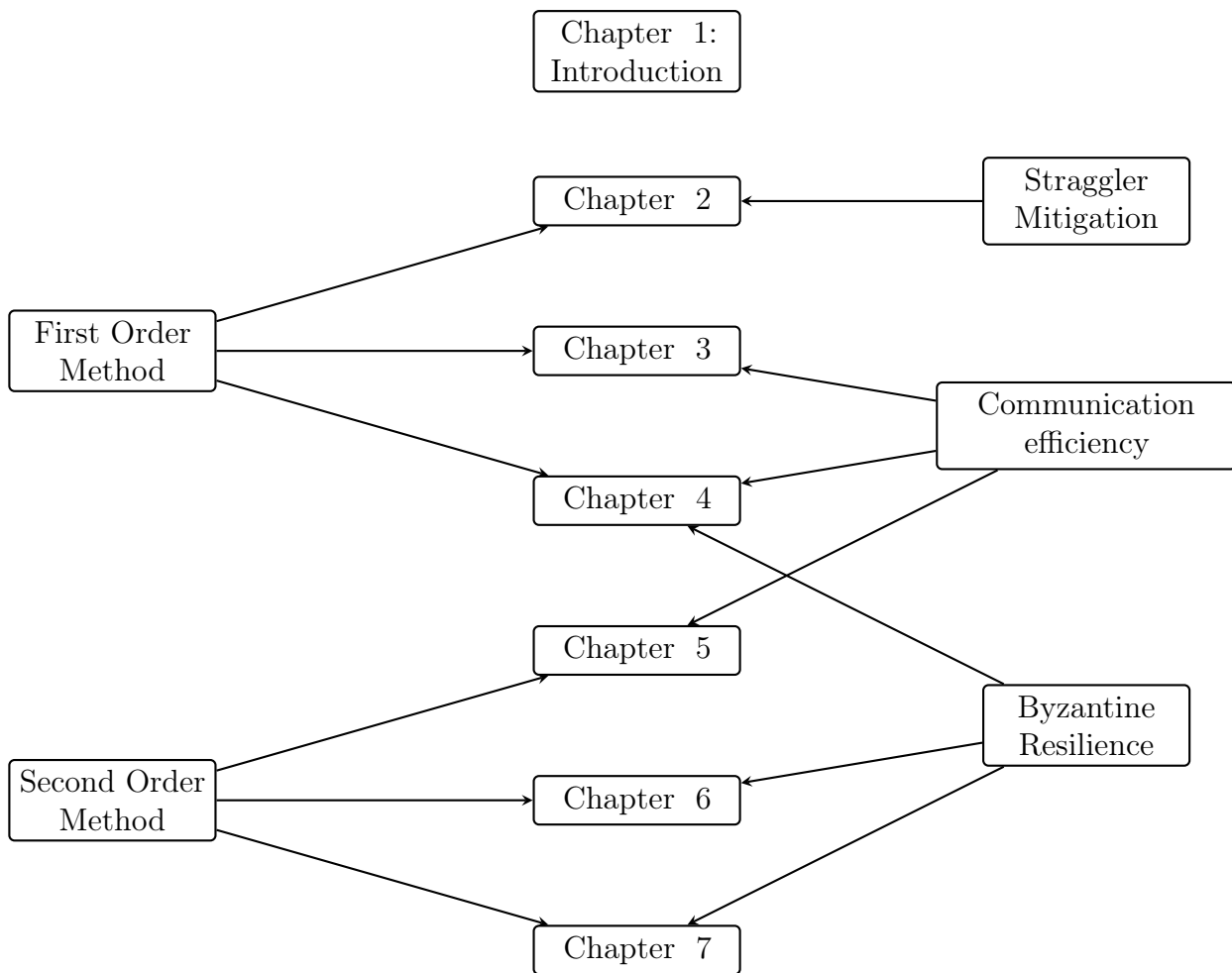
LIST OF FIGURES

Figure	Page
2.1 Total number of iterations and total computation time for solving the linear regression problem ($m = 2048$). The number of stragglers are 5 (top-left), 10 (top-right), 5 (bottom-left) and 10 (bottom-right).	33
2.2 Total number of iterations for solving sparse recovery problem in an overdetermined system ($m = 2048$). The left two figures correspond to the dimension 800 and the remaining ones correspond to dimension 1000. The number of stragglers are 5(top-left), 10 (top-right), 5 (bottom-left)and 10 (bottom-right).	34
2.3 Number of iterations and computation time for the sparse recovery problem in an underdetermined system ($k = 2000, m = 1024$). The number of stragglers are 5(top-left), 10 (top-right), 5 (bottom-left)and 10 (bottom-right).	35
3.1 Convergence for fully connected ReLU network compared to QSGD	71
3.2 Comparison of convergence for the least square problem with $d = 100, 200, 500$	72
3.3 Convergence of θ_t for $s = 1, 5, 10, 20$. for least square problem	73
4.1 Comparison of Robust Compressed Gradient Descent with and without thresholding scheme in a regression problem. The plots show better convergence with thersholding.	119
4.2 Comparison of Robust Compressed Gradient Descent with majority vote based <i>signSGD</i> [13] in regression Problem. The plots show better convergence with tthresholding in comparison to the majority vote based robestness of [13]	119
4.3 Comparison of norm based thresholding with and without error feedback. The plots show that error feedback based scheme offers better convergence.	121

4.4	Training (cross entropy) loss for MNIST image. Comparison with (a) Uncompressed Trimmed mean [135] (b) majority based <i>signSGD</i> of [13]. In plot (a) show that Robust Gradient descent matches the convergence of the uncompressed trimmed mean [135]. Plot (b) show a faster convergence compared to the algorithm of [13].	121
4.5	Training (cross entropy) loss for MNIST image. Different types of attack (a) labels with deterministic shift (9 – label) (b) random labels. Plots show theresholding scheme with different type of Byzantine attacks achieve similar convergence as ‘no Byzantine’ setup.	122
4.6	Convergence for (a) regression problem (b) training (cross entropy) loss for MNIST image. Plots show convergence beyond the theoretical bound on the number of Byzantine machine.	123
4.7	Convergence for (a) regression problem (b) training (cross entropy) loss for MNIST image. Plots show convergence with an negative Byzantine attack of $-\epsilon$ times the local gradient with high number of Byzantine machines for $\epsilon = 0.9$	123
5.1	Comparison of the convergence of DINGO and one and two round compression methods in terms of $\ \mathbf{g}_t\ $ (gradient norm) of regularized logistic regression for binary classification data.	147
6.1	(First row) Comparison of training accuracy between COMRADE(Algorithm 4) and GIANT [127] with (a) w5a (b) a9a (c) Epsilon (d) Covtype dataset. (Second row) Training accuracy of (e) GIANT for ‘flipped label’ and (f) ‘negative update’ attack; and comparison of Robust GIANT and COMRADE with a9a dataset for (g) ‘flipped label’ and (h) ‘negative update’ attack.	176
6.2	(First row) Accuracy of COMRADE with 10%, 15%, 20% Byzantine workers with ‘negative update’ attack for (a). w5a (b). a9a (c). covtype (d). Epsilon. (Second row) COMRADE accuracy with 10%, 15%, 20% Byzantine workers with ‘flipped label’ attack for (e) w5a (f) a9a (g) covtype (h) Epsilon. (Third row) Accuracy of COMRADE with ρ -approximate compressor (Section 6.4) with 10%, 15%, 20% Byzantine workers; (i) ‘flipped label’ attack for w5a (j) ‘negative update’ attack for w5a. (k) ‘flipped label’ attack for a9a . (l) ‘negative update’ attack for a9a dataset.	177

7.1	Function loss of the training data ‘a9a’ dataset (first row) and ‘w8a’ dataset (second row) with 10%, 15%, 20% Byzantine worker machines for (a,e). Flipped label attack.(b,f). Negative Update attack (c,g). Gaussian noise attack and (d,h). Random label attack for non-convex robust linear regression problem.	206
7.2	(a) Plot of the function value with different initialization to show that the algorithm escapes the saddle point with functional value 0. Comparison of our algorithm with ByzantinePGD [135] in terms of the total number of iterations to achieve small gradient norm ($\ \mathbf{g}\ \leq 0.1$) for (b) 10% (c) 15% and (d) 20% fraction Byzantine machines for different types of attack.	207
7.3	Classification accuracy of the testing data ‘a9a’ dataset (first row) and ‘w8a’ dataset (second row) with 10%, 15%, 20% Byzantine worker machines for (a,e). Flipped label.(b,f). Negative Update (c,g). Gaussian noise and (d,h). Random label attack for logistic regression problem.	208
7.4	(First row) Accuracy of the algorithm for logistic regression on test data of (a). a9a and (b). w8a dataset. (Second row). Function value of the non-convex robust linear regression on the training data of (a). a9a and (b). w8a dataset.	209

CHAPTER DEPENDENCIES



NOTATION

The following notation is used throughout the thesis unless otherwise specified.

- \triangleq reads “is defined to be equal to.”
- \mathbf{v} denotes a vector.
- \mathbf{M} denotes a matrix.
- \mathbb{F} denotes a generic field.
- \mathbb{R} denotes the real numbers.
- \mathbb{N} denotes the natural numbers (positive integers).
- For $n \in \mathbb{N}$, $[n]$ is the set $\{1, 2, \dots, n\}$.
- $\mathbf{0}_d$ and $\mathbf{1}_d$ denote the all zeros and all ones vectors in d -dimensions respectively.
- For $p \in \mathbb{N}$, $\|\mathbf{x}\|_p = (\sum_{i=1}^n x_i^p)^{1/p}$.
- S^{d-1} denotes the unit sphere.

CHAPTER 1

INTRODUCTION

The landscape of the modern technology is shaped and driven by machine learning and data science. With billions of smart devices and modern world applications such as self-driving cars, recommender system, genetics research etc, one of the most challenging part is to handle large scale data which makes training of sophisticated and powerful learning model extremely computationally-intensive and expensive. For example the OpenAI's GPT-3 language model has 175 billions model parameters requiring memory exceeding 350 GB and costing 12 millions USD to train¹. Even though such expensive and computationally intensive model is rare, the moderately larger machine learning systems are ubiquitous in modern data science industry. To cater to the computational need of such machine learning model and large volume of data, distributed learning setup (such as Amazon EC2) is used. A very general framework for distributed learning is worker and central or parameter server setup. The data is distributed among the worker nodes and being processed in parallel. The worker nodes then communicate the update based on the data locally stored to the central server where the final result is computed by aggregating the updates received. In practice, distributed system faces certain challenges in the form of delay due to straggler or slow worker, communication bottleneck due to high communication overhead between central server and worker machines, and disruption in the form of Byzantine failures in the worker machines. In this thesis, we address these problems

¹<https://venturebeat.com/2020/06/01/ai-machine-learning-openai-gpt-3-size-isnt-everything/>

and develop distributed first order (gradient based) and second order (Newton type) algorithms that are communication efficient and robust to such delays and faults.

In the rest of this chapter, we provide an overview of the thesis. We discuss the problem settings and our contributions that are considered in each chapter.

Straggler Mitigation : Generally in a distributed set-up (e.g., [36, 138]), the original large scale problems is divided into small problems and assigned to different worker nodes. The central (master) node collects the outcomes of the worker nodes and computes the results (potentially over multiple rounds of communication). In practical system, the process of collection of the outcomes from the worker nodes can be prone to unpredictable delays [35]. Such delays arise due to various reasons, including the slow-down at the workers and the congestion in the communication networks in the system. The workers that cannot provide the outcome of their local computation within a reasonable deadline due to these delays are termed *stragglers*. The presence of the stragglers can significantly degrade the performance of the system. Therefore, it is imperative that we address the variability in the response times of different components of the setup during the design of the computations tasks.

Multiple recent works explore the problem of mitigating the effect of stragglers. The replication schemes assign each task to multiple servers [104, 10, 124]. This ensures that the task gets completed without significant delay if at least one of the servers processing the task is non-straggler. In [84], Lee et al. explore the coding theoretic ideas that go beyond the replication schemes to address the issue of stragglers. In particular, they focus on linear computation, namely a matrix-vector product, and propose to encode the columns of the matrix by a maximum distance separable (MDS) code to obtain a taller encoded matrix. The rows of the encoded matrix are distributed among the workers, who are responsible for computing the inner product of the rows assigned to them with the vector in question. The redundancy among the rows of the encoded matrix allows for computation of the intended matrix-vector product even

if some of the servers fail to respond with the computation assigned to them. More exploratory works can be found in Dutta et al. [41], where sparsity is introduced in the rows of the matrix to make the computation faster. Other computational tasks (e.g., matrix-matrix product and convolution between vectors) have been explored in [86, 137, 42].

In Chapter 2, we consider the problem of data-fitting under square loss which is one of the most important loss function in machine learning, optimization and statistics. We employ *projected gradient descent* to solve the problem in distributed setup. To make the process robust against stragglers we employ low density parity check code (LDPC) to encode the second moment of the data which provide redundancy in computation of the gradient that is being carried out by multiple worker nodes in distributed manner. In each iteration, the master server collects the (partially) computed gradient by the worker servers and aggregates them to update the learning parameter. Due to the employment of LDPC in the pre-processing step the master server is able to compute good enough estimate of the gradient when the outcomes from the stragglers is not available. In Chapter 2 we discuss the setup, method, advantage of using LDPC in details and provide theoretical analysis and experimental results.

Communication Efficiency: Next, we focus on the problem of *communication bottleneck* in the distributed framework for large scale *stochastic gradient descent* (SGD) algorithm which is widely used in training large scale models. The distributed SGD model can easily be scaled on the basis of need but with high dimensional data, large scale model with millions of parameters and increasing number of servers as in Federated setup [77] the communication has become a *bottleneck* to the efficiency and speed of learning with SGD [28]. In the recent years, various quantization and sparsification techniques [3, 6, 12, 74, 90, 107, 118, 125, 131] have been developed to alleviate the problem of communication bottleneck. Recently, [66] even showed

the effectiveness of gradient quantization techniques for ReLU fitting. The goal of the quantization schemes is to efficiently compute either a low precision or a sparse unbiased estimate of the d -dimensional gradients. One also requires the estimates to have a bounded second moment in order to achieve guaranteed convergence.

In Chapter 3, we present a family of *vector-quantization* schemes that incur low communication costs while providing convergence guarantees. In particular, we provide explicit and efficient quantization schemes based on convex hull of specific structured point sets that require near optimal amount of communication necessary and sufficient for this purpose. At a high level, our scheme is based on the idea that any vector with bounded norm can be represented as convex combination of carefully constructed point set. Now with this construction, we can choose a point from the point set with probability proportional to its coefficient in the representation which makes it an unbiased estimator of the the given vector. For the purpose of communication, we can index the points in the point set in some order and communicate just the index. This process requires bits logarithmic of the size of the point set. We provide matching upper and lower bounds on this communication cost.

Moreover, the data samples used to train the model often contain sensitive information. Hence, preserving privacy of the participating clients is crucial. Differential privacy [43, 44] is a mathematically rigorous and standard notion of privacy considered in both literature and in practice. Informally, it ensures that the information from the released data (e.g. the gradient estimates) cannot be used to distinguish between two *neighboring* data sets. We provide construction with large convex hull where the variation in the coefficients of the convex combination of any two points of bounded norm is small. This kind of constructions provide privacy guarantee for certain range of ϵ which is the privacy parameter. For any general value of ϵ , we propose Randomized Response (RR) [130] and RAPPOR [45] based mechanisms that can be used over the proposed quantization with small trade-off in the variance of the estimates. The

instances of convex hull for the purpose of quantization is easy to construct and implement in practice.

Byzantine Resilience In First Order Learning : Next in Chapter 4, we focus on *Byzantine failure* which is another fundamental problem in the distributed learning setup. In many practical scenarios, messages or output from workers are susceptible to errors due to hardware faults or software bugs, stalled computations, data crashes, and unpredictable communication channels. In scenarios such as Federated Learning, users may as well be malicious and act adversarially. The inherent unpredictable (and potentially adversarial) nature of compute units is typically modeled as *Byzantine failures* ([79]). Even if a single worker is Byzantine, it can be fatal to most learning algorithms. In Chapter 3, we provide vector quantization for the purpose of the communication efficiency. In Chapter 4, we address both communication efficiency and Byzantine-robustness. Both these challenges have recently attracted significant research attention, albeit mostly separately. The problem of developing Byzantine-robust distributed algorithms has been considered in [7, 115, ?, 27, 135, 136, 15, 52]. And in Chapter 3, we discuss in details about the sparsification and quantization methods in terms of communication efficiency.

A notable exception to considering communication overhead separately from Byzantine robustness is the recent work of [13]. In this work, a sign-based compression algorithm *signSGD* of [12] is shown to be Byzantine fault-tolerant. The main idea of *signSGD* is to communicate the coordinate-wise signs of the gradient vector to reduce communication and employ a majority vote during the aggregation to mitigate the effect of Byzantine units. However, *signSGD* suffers from two major drawbacks. First, sign-based algorithms do not converge in general ([71]). In particular, [71, Section 3] presents several convex counter examples where *signSGD* fails to converge even though [13, Theorem 2] shows convergence guarantee for non-convex objective under certain assumptions. Second, *signSGD* can handle only a limited class of adversaries,

namely *blind multiplicative adversaries* ([13]). Such an adversary manipulates the gradients of the worker machines by multiplying it (element-wise) with a vector that can scale and randomize the sign of each coordinate of the gradient. However, the vector must be chosen before observing the gradient (hence ‘blind’).

For the communication efficiency, we employ δ -approximate compressor. For the adversarial model we restrict to an adversarial model in which Byzantine workers can provide arbitrary values as an input to the compression algorithm, but they correctly implement the mandated compression scheme. Though this adversarial model is restricted, we argue that it is well-suited for applications wherein compression happens outside of worker machines. Later we provide algorithm and analysis where we get rid of this restriction. To make the stochastic gradient descent algorithm Byzantine robust we simply discard the output (gradient from good nodes and arbitrary or possibly malicious value from Byzantine nodes) of a number of worker nodes (slightly higher than the number of Byzantine nodes believed to be in the setup) based on the norm of the output. This simple process provide good convergence property. We provide analysis and simulation results of our scheme. Furthermore, we strengthen our distributed learning algorithm by using error feedback to correct the direction of the local gradient. We show (both theoretically and via experiments) that using error-feedback [71] with a δ -approximate compressor indeed speeds up the convergence rate and attains better (statistical) error rate.

Sparse Distributed Approximate Newton : An alternative way to reduce the number of iterations (and hence the communication cost) between the workers and the central machine is to use second order optimization algorithms; which are known to converge much faster than their first order counterparts. A handful of algorithms has been developed using this philosophy, such as DANE [108], DISCO [140], GIANT [127], DINGO [32], Newton-MR [102], INEXACTDANE and AIDE [99]. However, the

question of whether it is possible to employ quantization/sparsification techniques in second order algorithms to further cut down communication cost remains unanswered.

In Chapter 5, we investigate and answer this aforementioned question affirmatively. Here, we use a δ -approximate compressor (same as used in Chapter 4) to provide communication efficiency in a recently proposed second order optimization algorithm DINGO [32]. We show different settings where we apply compression to reduce communication overhead in each iteration.

In each iteration of DINGO, the worker machines first communicate the local gradients to the central machine. The central machine aggregates the local gradients and broadcasts the global gradient to the worker machines. The worker machines then compute the local Hessians, obtain the (pseudo) inverse, compute the product of the inverse Hessian and the global gradient and send this vector to the central machine. Observe that, there are multiple places where compression can be employed. Such as:

1. *One round compression:* Every worker machine sends the uncompressed gradients to the central machine. However, while sending the product of the inverse (local) Hessian and gradient, it uses a δ -approximate compressor (as mentioned before), and sends the compressed vector. Hence, only one round of compression is employed in each iteration.
2. *Two round compression:* Here every worker compresses their local gradients and sends it to the central machine. Furthermore, while sending the product of local inverse Hessian and gradient, it also compresses that vector. So, each worker machine uses two rounds of compression (both with δ -approximate compressor) in each iteration.

We provide a careful analysis of the above settings such that we can track the effect of compression on the convergence rate. For setting 1, the conditions on the algorithm remain almost the same as that of DINGO with availability of the gradient

but differs in a ‘dot-product’ condition [32] as the second round of the communication (Hessian-gradient product) is compressed. For setting 2, we provide analysis with both rounds of communication being compressed while adhering to the same protocol of the underlying second order algorithm.

Byzantine Resilience In Second Order Learning : In the state-of-the-art distributed second order algorithms (like GIANT [127], DINGO [32], Determinantal Averaging [37]), which sequentially estimates functions of local gradients and Hessians and communicate them with the center machine. In sharp contrast with these algorithms, in Chapter 6, we propose COMRADE where the worker machines communicate *only once* per iteration with the center machine. We show that sequential estimation done by the previous approximate Newton type algorithm is redundant. In COMRADE, the worker machines only send a d dimensional vector, the product of the inverse of local Hessian and the local gradient. Via sketching arguments, we show that the empirical mean of the product of local Hessian inverse and local gradient is close to the global Hessian inverse and gradient product, and thus just sending the above-mentioned product is sufficient to ensure convergence. Hence, in this way, we save $\mathcal{O}(d)$ bits of communication per iteration. Furthermore, we argue that, in order to cut down further communication, the worker machines can even compress the local Hessian inverse and gradient product using the δ approximate compressor. In addition to the iteration reduction and compression for communication efficiency, we also make the algorithm resilient to Byzantine workers. For Byzantine resilience, COMRADE employs a simple thresholding policy on the norms of the local Hessian inverse and local gradient product. Since the norm of the Hessian-inverse and gradient product determines the *amount* of movement for Newton-type algorithms, this norm corresponds to a natural metric for identifying and filtering out Byzantine workers.

Escape Saddle Points: The standard and general approaches and convergence guarantees of the distributed learning algorithms (we discuss in Chapter 4 (first order method) and 6 (second order method)) work well for the convex loss function but provide weak results in the non-convex loss as a critical point in non-convex loss function may be a saddle point. [51, 72] shows that the stationary points of these problems are in fact saddle points and far away from any local minimum. Moreover, in [63, 116], it is argued that saddle points can lead to highly sub-optimal solutions in many problems of interest. This issue is amplified in high dimension as shown in [34], and becomes the main bottleneck in training deep neural nets. Hence designing efficient algorithm that escapes saddle points and find a local minima is of acute interest.

Furthermore, a line of recent work, shows that in many problems of interest, all local minima are global minima (e.g., dictionary learning [117], phase retrieval [116], matrix sensing and completion [14, 51], and some of neural nets [72]). Also, in [29], it is argued that for more general neural nets, the local minima are as good as global minima.

The issue of saddle point avoidance becomes non-trivial in the presence of Byzantine workers. Since we do not assume anything on the behavior of the Byzantine workers, it is certainly conceivable that by appropriately modifying their messages to the center, they can create *fake local minima* that are close to the saddle point of the loss function $f(\cdot)$, and these are far away from the true local minima of $f(\cdot)$. This is popularly known as the *saddle-point attack* (see [135]), and it can arbitrarily destroy the performance of any non-robust learning algorithm. Hence, our goal is to design an algorithm that escapes saddle points of $f(\cdot)$ in an efficient manner as well as resists the saddle-point attack simultaneously. The complexity of such an algorithm emerges from the the interplay between non-convexity of the loss function and the behavior of the Byzantine machines.

The problem of saddle point avoidance in the context of non-convex optimization has received considerable attention in the past few years. In the seminal paper of Jin et al. [64], a gradient descent based approach is proposed. By defining a certain *perturbation condition* and adding Gaussian noise to the iterates of gradient descent, the algorithm of [64] provably escapes the saddle points of the non-convex loss function. A few papers [133, 9] following the above use various modifications to obtain saddle point avoidance guarantees. However, these algorithms are non-robust. A Byzantine robust saddle point avoidance algorithm is proposed by Yin et al. [135], and probably is the closest to this work. In [135], the authors propose a repeated check-and-escape type of first order gradient descent based algorithm. First of all, being a first order algorithm, the convergence rate is quite slow (the rate for gradient decay is $1/\sqrt{T}$, where T is the number of iterations). Moreover, implementation-wise, the algorithm presented in [135] is computation heavy, and takes potentially many iterations between the center and the worker machines. Hence, this algorithm is not efficient in terms of the communication cost.

Here, we consider a variation of the famous cubic-regularized Newton algorithm of Nesterov and Polyak [94]. It is theoretically proved in [94] that a cubic-regularized Newton method with proper choice of parameters like step size always outperforms the gradient based first order schemes (like [135]) in all situations under consideration. We observe that the rate of gradient decay is $\frac{1}{T^{2/3}}$, which is strictly better than the first order gradient based methods.

CHAPTER 2

ROBUST GRADIENT DESCENT VIA MOMENT ENCODING AND LDPC CODES

We focus on the problem of fitting a structured linear model to the given data. In particular, given the features or data points $\{\mathbf{x}_i\}_{i \in [n] := \{1, \dots, n\}} \subset \mathbb{R}^d$ and the associated labels $\{y_i\}_{i \in [n]} \subset \mathbb{R}$, we want to learn the model parameter \mathbf{w}^* belonging to a structured set $\mathbf{w} \subset \mathbb{R}^d$ so that $y_i = \mathbf{x}_i^T \mathbf{w}^* + \epsilon_i$, for small modeling errors $\{\epsilon_i\}_{i \in [n]}$. In many applications, the prior knowledge about the structure of the model parameter (such as, sparsity and group sparsity) can be expressed with the help of a *regularizer* $\mathcal{R} : \mathbb{R}^d \rightarrow \mathbb{R}$ so that $\mathcal{W} \equiv \{\mathbf{w} \in \mathbb{R}^d : \mathcal{R}(\mathbf{w}) \leq R\}$, for $R \in \mathbb{R}$. In such settings, the task of recovering \mathbf{w}^* can be realized by solving the following optimization problem.

$$\min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{x}_i^T \mathbf{w})^2 \quad \text{subject to} \quad \mathbf{w} \in \mathcal{W} = \{\mathbf{w}' \in \mathbb{R}^d : \mathcal{R}(\mathbf{w}') \leq R\}. \quad (2.1)$$

Note that the square loss – employed in the optimization problem above – is one of the most pervasive loss functions in machine learning, optimization, and statistics. A large class of estimation problems arising in practice, such as compressed sensing [47], dictionary learning [89], and matrix completion [75], can be solved as special cases of the general optimization problem outlined in (2.1) [121].

Even though, we focus on the constrained optimization problem, our proposed solution easily extends to the unconstrained optimization problem

$$\min_{\mathbf{w} \in \mathbb{R}^d} \sum_{i=1}^m (y_i - \mathbf{x}_i^T \mathbf{w})^2 / 2 + \lambda \cdot \mathcal{R}(\mathbf{w}), \quad \mathcal{W} \in \{\mathbf{w}' \in \mathbb{R}^d : \mathcal{R}(\mathbf{w}') \leq R\}.$$

where we incorporate the regularizer in the objective function with the help of a regularization parameter $\lambda \in \mathbb{R}$.

We note that our proposed solution can also be employed to recover the structured model parameters for single-index or generalized linear models [65], where the given data fits a model of the form: $y_i = g(\mathbf{x}_i^T \mathbf{w}^*) + \epsilon_i$, with $g : \mathbb{R} \rightarrow \mathbb{R}$ denoting a possibly unknown nonlinear link function. In this setting, the model parameter can again be recovered by solving a generalized LASSO in (2.1) [96].

We employ the *projected gradient descent* (PGD) method to solve the underlying optimization problem. In a distributed computing setup, the iterative optimization procedure is implemented as follows. The master server maintains an estimate for the model parameter. In each step, the master sends the current estimate to the workers. The workers then compute a partial value of the gradient based on the received estimate and send the outcome of their computations to the master. By combining the messages received from the *non-straggling* workers, the master computes the gradient and updates its current estimate for the model parameter. In this chapter, our first contribution is to propose a preprocessing step which encodes the *second moment* of the data and then distributes the encoded moments among the workers. This way, there is some redundancy among the outcomes of the computation at the workers, which allows the master to obtain a good enough estimate of the gradient even when it does not receive the outcome of the computation assigned to the stragglers.

We employ the low-density parity check (LDPC) codes to encode the second moment of the data points. As a result, the task of calculating the gradient at the master reduces to the task of decoding an LDPC codeword in the presence of *erasures*, where the erased locations depend on the identities of the stragglers. The reason for working with LDPC codes is that the iterative decoding algorithms for these codes provide us three-fold benefits. The decoder has very low computational complexity and can automatically adjust to the number of the stragglers with a small number of

decoding iterations required if there are not too many stragglers present. Additionally, we can use the number of decoding iterations as a tuning parameter. Depending on the number of stragglers, we can run only those many decoding iterations that are sufficient to ensure the desired quality of the estimate of the gradient. In our setup, the number of erased coordinates of the gradient vector serves as a measure of its quality. Note that this measure is a *non-increasing* function of the number of decoding iterations. Finally, the MDS code based solutions provided in prior literature (such as, [84, 137]) suffer from the issue of noise-stability resulting from the low condition number of Vandermonde matrices, which we bypass by considering LDPC matrices.

Furthermore, we show that for a random model for stragglers, the PGD method with the proposed moment encoding scheme can be viewed as the projected stochastic gradient descent (PSGD) method. We then use the convergence analysis for the PSGD method to establish the convergence guarantee for our proposed solution. This analysis clearly characterizes the advantage over non-redundant or replication based gradient descent method in terms of the decoding iterations employed in each step of the method. We also conduct a detailed performance evaluation of our solution on a real-life distributed computing framework (swarm2) at the University of Massachusetts Amherst [119]. The performance results show that, as compared to the existing schemes, our proposed solution requires a smaller number of gradient steps in order to converge to the correct model parameter.

Furthermore, we show that for a random model for stragglers, the PGD method with the proposed moment encoding scheme can be viewed as the projected stochastic gradient descent (PSGD) method. We then use the convergence analysis for the PSGD method to establish the convergence guarantee for our proposed solution. This analysis clearly characterizes the advantage over non-redundant or replication based gradient descent method in terms of the decoding iterations employed in each step of the method. We also conduct a detailed performance evaluation of our solution on a

real-life distributed computing framework (swarm2) at the University of Massachusetts Amherst [119]. The performance results show that, as compared to the existing schemes, our proposed solution requires a smaller number of gradient steps in order to converge to the correct model parameter.

Comparison with other relevant works. In [84], Lee et al. focus on performing iterative gradient descent method in a distributed manner via repeatedly invoking their solution for coded computation of matrix-vector product. In this chapter, we also rely on the coded computation of matrix-vector product to realize iterative gradient descent in a straggler tolerant manner. However, we encode the second moment matrix as opposed to the plain data matrix as done in [84]. This leads to reduced communication rounds. Furthermore, this also makes the analysis of the optimization procedure completely different from that in [84]. As another novel contribution, we utilize LDPC codes which, as discussed above, allow for both efficient decoding and control over the quality of the (approximate) gradient computed in each step of the optimization procedure. In [67], Karakus et al. also study the problem of recovering the model parameter of a linear model by solving an alternative optimization problem where they encode both data points and their labels by the matrices with maximal (pairwise) incoherent columns. Again, our approach differs from theirs as we solve the original optimization problem itself and rely on moment encoding as opposed to data encoding.

In [120], Tandon et al. propose a novel framework, namely *gradient coding*, to counter the effect of stragglers on the performance of the gradient descent method. The gradient coding framework is designed for general loss functions which decompose over the data points. The gradient coding essentially relies on replication by cleverly assigning the data points to multiple workers to evaluate partial gradients. The specific designs for the replication among servers in the gradient coding framework along with their performance analysis are presented in [98, 24, 58]. Here, we note that the

square loss that we consider does have the additive structure. However, employing the (replication based) gradient coding framework to square loss leads to inefficient utilization of the compute and the communication resources. In [134], Yang et al. also study the iterative methods to solve linear inverse problems in the presence of stragglers. However, their setup significantly differs from our setup. In [134], multiple instances of the gradient descent method are run on different machines in a redundant manner such that each machine is responsible for locally solving an entire instance. Whereas in our setup, a single instance of the linear inverse problem is solved by a network of servers and each server communicates its partial results in each step of the gradient descent method. There exists a large literature dealing with various issues other than the stragglers in the context of distributed optimization and learning. We refer the readers to [84] for an excellent exposition of the literature.

Organization. We present the exact problem formulation along with the necessary background in Section 2.1. We present the main contribution of this chapter in Section 2.2 where we describe the moment encoding based optimization scheme along with its convergence analysis. In Section 2.3, we perform an extensive evaluation of the proposed scheme in a real-life distributed computing setup and compare it with the prior work. We present a list of notations in Table 2.1 for ease of reading.

2.1 System Model and Background

Our distributed computing setup has m worker servers and one master server. Performing large-scale computation in this setup involves dividing the desired computation problem into multiple small computation tasks that are assigned to the workers. The master then collects the outcomes of the tasks mapped to the workers and produces the final result. The overall computation may require multiple rounds of communication among the master and the workers.

Table 2.1: List of notation

n	Number of samples/data points
d	Dimension of samples
(N, K)	Length and dimension of the employed code
$l = N \frac{d}{K}$	Length of the encoded vectors
m	Number of worker servers
$\alpha = \frac{l}{m}$	Number of rows mapped to each server
\mathbf{w}	Model parameter to be learnt
$\ell((y, \mathbf{x}), \mathbf{w})$	Loss associated with \mathbf{w} for data point \mathbf{x} and label y
$\mathcal{L}(\mathbf{w})$	Total empirical loss associated with \mathbf{w}
k	Index for LDPC decoding iterations
D	Number of iteration of LDPC decoding during each gradient descent step
t	Index for gradient descent steps
T	Number of gradient descent steps
η_t	Learning rate for gradient descent
\mathcal{S}_t	Set of servers available during the t -th gradient descent step
s	Number of stragglers

We are given n data samples or feature vectors $\{\mathbf{x}_i\}_{i \in [n]} \subset \mathbb{R}^d$ and their labels $\{y_i\}_{i \in [n]} \subset \mathbb{R}$. We are mainly concerned with learning a structured linear model. In particular, we are interested in learning a vector $\mathbf{w}^* \in \mathcal{W} \equiv \{\mathbf{w}' \in \mathbb{R}^d : \mathcal{R}(\mathbf{w}') \leq R\}$, for some regularizer $\mathcal{R} : \mathbb{R}^d \rightarrow \mathbb{R}$, such that the following total empirical loss is minimized. In this work, we are mainly concerned with the following linear regression task, where we are interested in learning a vector $\mathbf{w} = (w_1, w_2, \dots, w_d) \in \mathbb{R}^d$ with $\mathcal{R}(\mathbf{w}) \leq R$, for some regularizer $\mathcal{R} : \mathbb{R}^d \rightarrow \mathbb{R}$, such that the following total empirical loss function is minimized.

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 = \frac{1}{2} (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) = \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{x}_i^T \mathbf{w})^2, \quad (2.2)$$

where $\mathbf{y} = (y_1, y_2, \dots, y_n)^T \in \mathbb{R}^m$ and $\mathbf{X} = (\mathbf{x}_1 \ \mathbf{x}_2 \ \dots \ \mathbf{x}_n)^T \in \mathbb{R}^{n \times d}$. Note that the gradient of the total empirical loss with respect to \mathbf{w} has the following form.

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = (\mathbf{X}^T \mathbf{X} \mathbf{w} - \mathbf{X}^T \mathbf{y}) = \sum_{i=1}^n (\mathbf{x}_i \mathbf{x}_i^T \mathbf{w} - y_i \mathbf{x}_i). \quad (2.3)$$

In this paper, we rely on the PGD method to solve the underlying constrained optimization problem, which iteratively updates an estimate of \mathbf{w}^* . Specifically, at the t -th step, the estimate \mathbf{w}_t has the form

$$\mathbf{w}_t = P_{\mathbf{w}}(\mathbf{w}_{t-1} - \eta_t \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}_{t-1})), \quad (2.4)$$

where η_t is the learning rate at the t -th step, which may potentially be independent of t ; and the projection operator $P_{\mathbf{w}} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is defined to be $\mathbf{w} \mapsto \arg \min_{\tilde{\mathbf{w}} \in \mathcal{W}} \|\mathbf{w} - \tilde{\mathbf{w}}\|_2^2$.

Remark 2.1. *In our proposed scheme, the master performs the projection step in (2.4). Thus, we are mainly interested in the regularizers with computationally efficient projection operations. This is particularly true for decomposable regularizers, such as sparsity constraints.*

Preliminary: Linear codes. We rely on linear codes to perform the overall computation on a distributed computing setup in redundant manner. The redundancy allows the master to realize the original computation task in a straggler tolerant manner. An (n, k) linear code is simply a subspace of dimension k belonging to an n -length vector space. Here, we focus on the vector space defined over the real numbers \mathbb{R} . Therefore, an (n, k) linear code \mathcal{C} forms a k -dimensional subspace in \mathbb{R}^n . Given an k -length message vector $\mathbf{x} \in \mathbb{R}^k$, it can be *encoded* (or mapped) to a *codeword* from the code \mathcal{C} with the help of a generator matrix $\mathbf{G} \in \mathbb{R}^{n \times k}$ as $\mathbf{c} = \mathbf{G}\mathbf{x} \in \mathcal{C}$. Thus, a linear code can be defined as $\mathcal{C} := \{\mathbf{c} \in \mathbb{R}^n : \mathbf{c} = \mathbf{G}\mathbf{x} \text{ for some } \mathbf{x} \in \mathbb{R}^k\}$. Alternatively,

a linear code can also be defined by a parity check matrix $\mathbf{H} \in \mathbb{R}^{(n-k) \times n}$ as follows $\mathcal{C} := \{\mathbf{c} \in \mathbb{R}^n : \mathbf{H}\mathbf{c} = \mathbf{0}\}$. A generator matrix leads to a *systematic* encoding, if for each $\mathbf{x} \in \mathbb{R}^k$, the message vector \mathbf{x} exactly appears as k coordinates of the associated codeword $\mathbf{c} = \mathbf{G}\mathbf{x}$. The redundancy introduced by mapping a k dimensional vector \mathbf{x} to an n -dimensional vector \mathbf{c} with $n > k$ allows one to recover \mathbf{x} from \mathbf{c} even when some of the coordinates of \mathbf{c} are missing. In particular, if the code \mathcal{C} has minimum distance d_{\min} , then \mathbf{x} can be recovered even if any $d_{\min} - 1$ coordinates of \mathbf{c} are not available.

2.1.1 The Data Coding Method of [84] and the Gradient Coding Approach of [120]

An approach to run gradient descent in a distributed system using reliable distributed matrix multiplication as a building block was recently presented by Lee et al. [84]. Note that, in the linear regression problem, computing the gradient of the total empirical loss involves computation of two matrix-vector products in each iteration (see (2.3)), namely: $X\mathbf{w}_{t-1}$ and $\mathbf{X}^T(\mathbf{X}\mathbf{w}_{t-1} - \mathbf{y})$. In [84], an MDS-coded distributed algorithm for matrix multiplication was proposed. In this algorithm, to perform the matrix-vector product $\mathbf{A}\mathbf{x}$, the matrix \mathbf{A} is premultiplied by the generator matrix \mathbf{G} of an MDS code of proper dimensions to get $\tilde{\mathbf{A}} = \mathbf{G}\mathbf{A}$. Each worker node then performs a single inner product (or a set of inner products) involving a row of $\tilde{\mathbf{A}}$ and \mathbf{x} . The results of these local computations are then sent to the master node. As long as the number of workers that successfully deliver their local computations within the deadline is more than a specified threshold (in other words, as long as the number of stragglers is within the erasure correcting capability of the MDS code given by \mathbf{G}), the product $\mathbf{A}\mathbf{x}$ can be found at the master node. In each iteration of the gradient descent, the above matrix-vector product protocol is applied twice (see [84] for details)

to compute $\mathbf{X}\mathbf{w}_{t-1}$ and $\mathbf{X}^T(\mathbf{X}\mathbf{w}_{t-1} - \mathbf{y})$. This facilitates computation of the gradient in each iteration in the presence of the stragglers.

In [120], Tandon et al. propose a novel framework to exactly compute gradient of the underlying loss function in a distributed computation setup. In particular, they consider a generic loss function which takes the following additive form.

$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^m \ell((y_i, \mathbf{x}_i), \mathbf{w})$. For such a loss function, its gradient can be obtained as

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \sum_{i=1}^m \nabla_{\mathbf{w}} \ell((y_i, \mathbf{x}_i), \mathbf{w}). \quad (2.5)$$

In order to compute the gradient in a distributed manner, the samples and the corresponding labels are distributed among w workers in a redundant manner. For $i \in [m]$, the samples and labels allocated to the i -th worker server are indexed by the set $\mathcal{A}_i \subseteq [m]$. Given the samples and labels indexed by the set \mathcal{A}_i , the i -th worker can compute the following components of the gradient (cf. (2.5)).

$$\mathcal{B}_i := \{ \nabla_{\mathbf{w}} \ell((y_j, \mathbf{x}_j), \mathbf{w}) \}_{j \in \mathcal{A}_i} \subset \mathbb{R}^d. \quad (2.6)$$

Now, the i -th worker transmits a linear combination of the blocks in \mathcal{B}_i to the master. In particular, the transmitted block can be represented as follows.

$$\mathbf{z}_i = \sum_{j \in \mathcal{A}_i} b_{i,j} \nabla_{\mathbf{w}} \ell((y_j, \mathbf{x}_j), \mathbf{w}) \in \mathbb{R}^d. \quad (2.7)$$

Equivalently, the transmitted blocks from all m workers can be represented as the following $m \times d$ matrix.

$$\mathbf{Z} = (\mathbf{z}_1, \dots, \mathbf{z}_m)^T = \mathbf{B}(\nabla_{\mathbf{w}} \ell((y_1, \mathbf{x}_1), \mathbf{w}) \cdots \nabla_{\mathbf{w}} \ell((y_d, \mathbf{x}_d), \mathbf{w}))^T,$$

where \mathbf{B} is an $m \times d$ matrix containing the coefficients associated with the transmission from m workers (cf. (2.7)). Note that, for $i \in [m]$, the support of the i -th row of the matrix \mathbf{B} is contained in the set \mathcal{A}_i .

Let $\mathcal{S} \subset [m]$ denote the set of indices of the workers that successfully deliver their local computations within the deadline. Assuming that we have s straggling workers which do not respond with their intended transmission before the deadline, we have $|\mathcal{S}| = m - s$. Note that the master has following information at its disposal.

$$\mathbf{Z}_{\mathcal{S}} = \mathbf{B}_{\mathcal{S}} (\nabla_{\mathbf{w}} \ell((y_1, \mathbf{x}_1), \mathbf{w}) \cdots \nabla_{\mathbf{w}} \ell((y_m, \mathbf{x}_m), \mathbf{w}))^T, \in \mathbb{R}^{(m-s) \times d}, \quad (2.8)$$

where $\mathbf{Z}_{\mathcal{S}}$ and $\mathbf{B}_{\mathcal{S}}$ denote the sub-matrices formed by the rows indexed by \mathcal{S} in \mathbf{Z} and \mathbf{B} , respectively. In order to be able to obtain the gradient

$$\begin{aligned} \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) &= \sum_{i=1}^n \nabla_{\mathbf{w}} \ell((y_i, \mathbf{x}_i), \mathbf{w}) \\ &= (1, \dots, 1) \cdot (\nabla_{\mathbf{w}} \ell((y_1, \mathbf{x}_1), \mathbf{w}) \cdots \nabla_{\mathbf{w}} \ell((y_n, \mathbf{x}_n), \mathbf{w}))^T, \end{aligned}$$

we require that the all ones vector $(1, \dots, 1)$ belongs to the subspace spanned by the rows of the matrix $\mathbf{B}_{\mathcal{S}}$. *Therefore, the design criterion in the gradient coding approach [120] is to find an allocation of the samples $\{\mathcal{A}_i\}_{i \in [s]}$ and the associated transmission matrix \mathbf{B} such that for every $\mathcal{S} \subset [s]$ with $|\mathcal{S}| = m - s$, all ones vector $(1, \dots, 1)$ belongs to the row-space of the matrix $\mathbf{B}_{\mathcal{S}}$.*

Our computing method crucially differs from both of the schemes of [120] and [84]. Instead of encoding the matrices \mathbf{X} and \mathbf{X}^T with MDS codes we use a single code to encode the matrix $\mathbf{X}^T \mathbf{X}$, the second moment of the data.

2.2 Encoding Second Moment : Optimization with Approximate Gradient

We exploit the special structure of the gradient of the square loss (cf. (2.3)) to devise a scheme to deal with stragglers. The proposed scheme is more efficient as compared to the gradient coding approach [120] and the reliable distributed matrix multiplication based scheme [84]. Recall the gradient of the total empirical loss associated with the square loss function from (2.3). Note that we need to compute the term $\mathbf{X}^T \mathbf{y}$ only once at the beginning of the optimization procedure as it is independent of the optimization parameter \mathbf{w} . By using the notation $\mathbf{M} = \mathbf{X}^T \mathbf{X}$ and $\mathbf{b} = \mathbf{X}^T \mathbf{y}$, the t -th step of the PGD method takes the following form (cf. (2.4)).

$$\mathbf{w}_t = P_{\mathbf{w}}(\mathbf{w}_{t-1} - \eta_t \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}_{t-1})) = P_{\mathbf{w}}(\mathbf{w}_{t-1} - \eta_t (M \mathbf{w}_{t-1} - \mathbf{b})). \quad (2.9)$$

where \mathbf{w}_t denotes the estimate of \mathbf{w}^* at the end of t -th step.

2.2.1 Exact Computation of Gradient in Each Step

Now, in order to perform the projected gradient descent in a distributed computation setup, we distribute the task of computing matrix-vector product $\mathbf{M} \mathbf{w}_t$ among the m workers. In particular, we encode the $d \times d$ matrix \mathbf{M} using a linear code. The encoded matrix can be used to generate redundant tasks for workers which subsequently enable us to mitigate the effect of stragglers.

Scheme 2.1 (Exact gradient computation using linear codes:). *Given the matrix $\mathbf{M} = \mathbf{X}^T \mathbf{X}$ and an $(N = m, K)$ linear code¹ \mathbf{C} , the gradient computation for each step of the optimization procedure is realized as below.*

¹For the ease of exposition, we assume that K divides d .

- Let $\mathbf{m}_1, \dots, \mathbf{m}_d$ denote the d rows of the matrix $\mathbf{M} = \mathbf{X}^T \mathbf{X}$. Let $\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_{d/K} \subset [d]$ represent a partition of the set indices for these rows $[d]$ such that $\mathcal{P}_i \cap \mathcal{P}_j = \emptyset$ for $i \neq j$ and $|\mathcal{P}_i| = K \ \forall i \in [d/K]$.
- For each $i \in [d/K]$, we encode the $K \times d$ matrix $\mathbf{M}_{\mathcal{P}_i}$ using the $(N = m, K)$ linear code \mathcal{C} as

$$\mathbf{C}^{(i)} = \mathbf{G} \mathbf{M}_{\mathcal{P}_i} \in \mathbb{R}^{N \times d}, \quad (2.10)$$

where \mathbf{G} is an $N \times K$ generator matrix of \mathcal{C} . Note that the d columns of the matrix $\mathbf{C}^{(i)}$ form d codewords of \mathcal{C} .

- In the distributed computation setup, for $i \in [d/K]$ and $j \in [N] = [m]$, we now allocate j -th row of $\mathbf{C}^{(i)}$ to the j -th worker. This way, the j -th server is assigned the following sets of $\alpha = \frac{N}{m} \cdot \frac{d}{K} = \frac{d}{K}$ vectors.

$$\mathcal{T}_j = \{\mathbf{c}_j^{(1)}, \dots, \mathbf{c}_j^{(\frac{d}{K})}\} \subset \mathbb{R}^d, \quad (2.11)$$

where $\mathbf{c}_j^{(i)}$ denotes the j -th row of the matrix $\mathbf{C}^{(i)}$.

- During the t -th step of the gradient descent optimization procedure, j -th worker is tasked with computing the inner product of the rows assigned to it with the current estimate \mathbf{w}_{t-1} , i.e., the j -th worker sends $\alpha = \frac{d}{K}$ inner products

$$\{\langle \mathbf{c}_j^{(1)}, \mathbf{w}_{t-1} \rangle, \dots, \langle \mathbf{c}_j^{(\frac{d}{K})}, \mathbf{w}_{t-1} \rangle\} \quad (2.12)$$

to the master.

- **Straggler tolerant exact gradient computation:** Assuming that the workers indexed by the set $\mathcal{S}_t^C := [m] \setminus \mathcal{S}_t$ behave as stragglers during the t -th step of

the optimization procedure, the master has access to the following information received from the non-straggling workers.

$$\mathbf{C}_{S_t}^{(i)} \mathbf{w}_{t-1} = \mathbf{G}_{S_t} \mathbf{M}_{P_i} \mathbf{w}_{t-1} \text{ for all } i \in [d/K]. \quad (2.13)$$

Since the code \mathcal{C} generated by \mathbf{G} is a linear code, it's straightforward to verify that for each $i \in [d/K]$, $\mathbf{C}^{(i)} \mathbf{w}_{t-1} = \mathbf{G} \mathbf{M}_{P_i} \mathbf{w}_{t-1}$ corresponds to a codeword of \mathcal{C} . Moreover, the information available at the master (cf. (2.13)) is equivalent to observing these codewords with some of their coordinates erased. Assuming the code \mathcal{C} has large enough minimum distance, or equivalently, the matrix \mathbf{G}_{S_t} is full rank, the master can recover $\mathbf{M}_{P_1} \mathbf{w}_{t-1}, \dots, \mathbf{M}_{P_{d/K}} \mathbf{w}_{t-1}$ from the information received from the workers indexed by the set S_t . This allows the master to construct $\mathbf{M} \mathbf{w}_{t-1} = \mathbf{X}^T \mathbf{X} \mathbf{w}_{t-1}$ and update the estimate for \mathbf{w} according to (2.9).

We now state the following result about the performance of Scheme 2.1, which follows from the description of the scheme in a straightforward manner.

Proposition 2.2. *Assume that the moment encoding based Scheme 2.1 employs an $(N = m, K)$ linear code \mathcal{C} with minimum distance d_{\min} . Then, the scheme implements exact gradient descent method as long as the number of the stragglers during each step of the optimization is strictly less than d_{\min} .*

Remark 2.2. *Note that length of the code \mathcal{C} does not need to be equal to the number of workers. For the ease of exposition, we focus on the $N = m$ case here. This choice provides a simple natural allocation of computation tasks to the workers. However, suitable allocation can also be devised for the setting with $N \neq d$.*

Comparison with Gradient Coding Approach [120]. Encoding the second moments offers an immediate advantage over the general gradient coding approach for the underlying optimization problem (cf. (2.1)). In Scheme 1, during a step

of the optimization method, each worker communicates one scalar for each of the rows assigned to it. Whereas, in gradient coding, each worker needs to transmit a d -dimensional vector to the master. Moreover, as for the local computation at a worker during each step, our approach requires computing a single inner product for every row assigned to the worker. In contrast, in the gradient coding framework, workers have to perform matrix-vector products between $d \times d$ rank 1 matrices and d -dimensional vectors.

In Scheme 2.1, we employ linear codes with the objective that the master should be able to compute (decode) the exact gradient during every step of the optimization procedure. This can be achieved by utilizing any linear code with large enough minimum distance. However, for the PGD method to succeed, it's not necessary to compute the exact gradient in every step. In particular, the stochastic gradient descent method is one of the most used versions of the gradient descent methods, where one employs an estimate of the gradient based on a randomly chosen sample and its label [105]. For the problem at hand, the t -th step of projected stochastic gradient descent (PSGD) method is as follows.

$$\mathbf{w}_t = P_{\mathbf{w}}(\mathbf{w}_{t-1} - \eta_t \cdot n \cdot (\mathbf{x}_i \mathbf{x}_i^T \mathbf{w}_{t-1} - y_i \mathbf{x}_i)), \quad (2.14)$$

where i denotes an integer that is picked uniformly at random from $[n]$. Note that $n \cdot (\mathbf{x}_i \mathbf{x}_i^T \mathbf{w} - y_i \mathbf{x}_i)$ indeed gives an unbiased estimate of the true gradient (cf. (2.3)) as $n \cdot \mathbb{E}[(\mathbf{x}_i \mathbf{x}_i^T \mathbf{w} - y_i \mathbf{x}_i)] = \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w})$. Next, we exploit this robustness of the gradient based procedures to the quality of the gradient.

2.2.2 Approximate Recovery of Gradient in Every Step

Here, we focus on implementing the gradient based optimization procedure in a distributed computing setup by constructing only an estimate of the true gradient during each step of the optimization procedure. This allows us to employ coding

schemes that have low complexity encoding and decoding algorithms, which lowers the overall computational complexity of the coding based approach to mitigate the effect of stragglers. Before we describe our approximate gradient based optimization procedure, we specify the assumptions on the identity of the stragglers during each step of the optimization procedure.

Assumption 2.1 (Straggling behavior of the workers). *Let the indices of the stragglers $\mathcal{S}_t^C \subset [m]$ during the t -th step of the optimization be distributed independent of the stragglers in the previous steps. Furthermore, let the distribution of the stragglers in each step be such that each worker independently behaves as a straggler with probability q_0 .*

The analysis of this section can be modified for the other random models for the identity of the stragglers. Here, we note that we do not ensure any such random model for the straggling behavior during our experimental evaluations of the proposed scheme in Section 2.3.

We are now in the position to describe the LDPC codes based optimization procedure that rely on approximate gradient during each step of the optimization procedure.

Scheme 2.3 (LDPC codes based optimization with approximate gradients). *Given the matrix $\mathbf{M} = \mathbf{X}^T \mathbf{X}$, we take an $(N = m = d + p, K = d)$ LDPC code \mathcal{C} with $\mathbf{H} \in \mathbb{R}^{p \times N}$ as its (low-density) parity check matrix.² The approximate gradient based optimization procedure is realized as follows.*

- *Encode $\mathbf{M} = \mathbf{X}^T \mathbf{X}$ using a systematic matrix of \mathcal{C} , say \mathbf{G} , as $\mathbf{C} = \mathbf{GM}$, where without loss of generality we assume that \mathbf{M} constitutes the first d rows of the*

²For the ease of exposition, in addition to $N = m$, we assume that $K = d$. The proposed scheme can be easily generalized to the setting with $d > K$, as done in Scheme 2.1 by partitioning the rows of \mathbf{M} in the blocks of K rows.

matrix \mathbf{C} . Next, distribute the $m = d + p$ rows of \mathbf{C} among m workers such that the j -th row \mathbf{c}_j is assigned to the j -th worker.

- During the t -th step of the optimization procedure, j -th worker computes the inner product of the row assigned to it with the current estimate \mathbf{w}_{t-1} and sends $\mathbf{c}_j^{(1)} \mathbf{w}_{t-1} \in \mathbb{R}$ to the master.
- Assuming that the set $\mathcal{S}_t^C := [m] \setminus \mathcal{S}_t$ denotes the indices of stragglers during t -th step, the information received at the master takes the form:

$$\mathbf{C}_{\mathcal{S}_t} \mathbf{w}_{t-1} = \mathbf{G}_{\mathcal{S}_t} \mathbf{M} \mathbf{w}_{t-1}. \quad (2.15)$$

Note that $\mathbf{c} = \mathbf{G} \mathbf{M} \mathbf{w}_{t-1}$ is a codeword of \mathcal{C} with $\mathbf{M} \mathbf{w}_{t-1}$ appearing in its first d coordinates.

- **Computation of approximate gradient:** Given $\mathbf{c}_{\mathcal{S}_t} = \mathbf{C}_{\mathcal{S}_t} \mathbf{w}_{t-1} = \mathbf{G}_{\mathcal{S}_t} \mathbf{M} \mathbf{w}_{t-1}$, the master employs D iterations of an iterative erasure correction algorithm for the LDPC code \mathcal{C} , where \mathcal{S}_t^c denotes the indices of the erased coordinates. Let $\hat{\mathbf{c}}(t; D) = (\hat{c}(t; D)_1, \dots, \hat{c}(t; D)_d)$ be the estimate for the codeword \mathbf{c} after D iterations of the erasure correction algorithm [100]. If a particular coordinate is not recovered by the end of D iterations, we replace the coordinate with 0. Let $\mathcal{U}_t \subseteq [d]$ denote the set of indices of the coordinates that are set to 0 in this manner. Subsequently, we construct a vector $\hat{\mathbf{b}}_t$ by setting those coordinates of $\mathbf{b} = \mathbf{X}^T \mathbf{y}$ to 0 that are in \mathcal{U}_t . During the t -th step, the master updates the current estimate of \mathbf{w}^* as

$$\mathbf{w}_t = P_{\mathbf{w}} \left(\mathbf{w}_{t-1} - \eta_t \cdot \left((\hat{c}(t; D)_1, \dots, \hat{c}(t; D)_d)^T - \hat{\mathbf{b}}_t \right) \right). \quad (2.16)$$

In what follows, we establish that under Assumption 2.1, Scheme 2.3 indeed implements a variant of the PSGD method. As a result, under some natural requirements on the loss function and the initialization \mathbf{w}_0 , we obtain a convergence result for

Scheme 2.3 that is similar to those available in the literature for the PSGD method (cf. (2.14)).

However, before we analyze the convergence of Scheme 2.3, we need to characterize the quality of the gradient recovered at the end of D iterations of the erasure correction algorithm of the underlying LDPC code \mathcal{C} . The LDPC codes have been extensively studied in the literature along with the performances of various decoding algorithms for such codes [48, 111, 100]. Under Assumption 2.1, where each worker independently behaves as a straggler with probability q_0 , the vector received by the master (cf. (2.15)) is equivalent to the outcome of an erasure channel. For a specific family of LDPC codes and a fixed iterative erasure correction algorithm, there have been many successful attempts to characterize the likelihood of an initially erased coordinate being recovered after a certain number of iterations. Here, we state a special case³ of the most prominent result in this direction which applies to various random ensembles of LDPC codes with sufficiently large length. This results is obtained by density evolution analysis [100].

Proposition 2.4. *Consider an ensemble of LDPC code defined by the random $p \times N$ parity check matrix \mathbf{H} such that each of the p rows (N columns) of the matrix \mathbf{H} have z (r) nonzero entries.⁴ Let each coordinate of a codeword from the ensemble be independently erased with the probability q_0 . Then, the probability q_k that a coordinate of the codeword remains erased after k iterations of the iterative erasure correction satisfies the relationship⁵ $q_k = q_0 \cdot (1 - (1 - q_{k-1})^{r-1})^{z-1}$.*

³In particular, we restrict ourselves to the LDPC codes with left and right regular Tanner graphs. We refer the readers to [100] for the general version of the result that applies to LDPC codes with irregular Tanner graphs.

⁴There are multiple ways of generating a random ensembles of LDPC codes (see e.g., [100][Ch. 3]).

⁵The relation in here is shown to hold with very high probability, which involves application of bounded-difference concentration inequality on the random bipartite graphs corresponding to \mathbf{H} . Given that these are fairly standard results in the coding theory literature, we refer the readers to [101, 100] for the details.

Remark 2.3. *The key take away from Proposition 2.4 is that the probability of a coordinate of a codeword staying erased is a monotonically non-increasing function of the number of iterations as long as $q_0 < q^*(r, z) < 1$, where $q^*(r, z)$ is function of the row and column weights of the random matrix \mathbf{H} .*

The following lemma characterizes the quality of the gradient vector obtained at the master after D iterations of the erasure correction algorithm of the underlying LDPC code.

Lemma 2.5. *Let the distribution of stragglers satisfy Assumption 2.1 and the master node employs D iterations of the erasure correction algorithm. Then, during t -th step of the optimization procedure, we have*

$$\mathbb{E} \left[(\hat{c}(t; D)_1, \dots, \hat{c}(t; D)_d)^T - \hat{\mathbf{b}}_t \right] = (1 - q_D) \cdot \nabla \mathcal{L}(\mathbf{w}_{t-1}),$$

which is a scaled version of the true gradient at \mathbf{w}_{t-1} .

Proof of lemma 2.5. Recall that, during the t -th step of the optimization procedure, q_D denotes the probability that a particular coordinate of the codeword $\mathbf{c} = \mathbf{C}\mathbf{w}_{t-1} \in \mathbb{R}^N$ is not recovered by the master (cf. Scheme 2.3). The first d coordinates of this vector correspond to the true gradient vector at \mathbf{w}_{t-1} . Therefore, for $i \in [d]$, we have

$$\mathbb{P} [\hat{c}(t; D)_i = c_i] = 1 - q_D \text{ and } \mathbb{P} [\hat{c}(t; D)_i = 0] = q_D. \quad (2.17)$$

Similarly, for $i \in [d]$, we have,

$$\mathbb{P} [\hat{b}_i = b_i] = 1 - q_D \text{ and } \mathbb{P} [\hat{b}_i = 0] = q_D. \quad (2.18)$$

By using (2.17) and (2.18), it is straightforward to verify that

$$\begin{aligned}\mathbb{E} \left[(\hat{c}(t; D)_1, \dots, \hat{c}(t; D)_k)^T - \hat{\mathbf{b}}_t \right] &= (1 - q_D) \cdot ((c_1, \dots, c_k)^T - \mathbf{b}) \\ &\stackrel{(i)}{=} (1 - q_D) \cdot (\mathbf{M}\mathbf{w}_{t-1} - \mathbf{X}^T \mathbf{y}) \\ &= (1 - q_D) \cdot \nabla \mathcal{L}(\mathbf{w}_{t-1}),\end{aligned}$$

where (i) follows from the systematic form associated with the generator matrix G . □

Convergence Analysis of Scheme 2.3 Here, we formally argue that the proposed Scheme 2.3 enjoys the convergence guarantees similar to those available for the typical PSGD method. In fact, the proof of the convergence of our scheme heavily relies on the ideas employed in the proof of convergence for PSGD algorithm as described in [93]. Recall that the total empirical loss associated with the model parameter $\mathbf{w} \in \mathbb{R}^d$ for given set of data samples $\{\mathbf{x}_i\}_{i \in [n]} \subset \mathbb{R}^d$ and the corresponding labels $\{y_i\}_{i \in [n]} \subset \mathbb{R}$ takes the form.

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^n \ell((y_i, \mathbf{x}_i), \mathbf{w}) = \frac{1}{2} \cdot \sum_{i=1}^n (y_i - \mathbf{x}_i^T \mathbf{w})^2. \quad (2.19)$$

We now state the convergence result for Scheme 2.3 which holds under natural assumptions on the loss function and the initialization for the optimization procedure \mathbf{w}_0 . In what follows we use $\|\cdot\|$ to denote the ℓ_2 norm $\|\cdot\|_2$. We also note that the projection operator $P_{\mathbf{w}}$ is non-expanding, i.e.,

$$\|P_{\mathbf{w}}(\mathbf{w}) - P_{\mathbf{w}}(\mathbf{w}')\| \leq \|\mathbf{w} - \mathbf{w}'\| \quad \text{for all } \mathbf{w}, \mathbf{w}' \in \mathbb{R}^d.$$

Theorem 2.6. *Suppose for all $(\mathbf{x}, y) \in \mathbb{R}^{d+1}$ and $\mathbf{w} \in \mathcal{W}$, the loss function satisfies $\|\nabla \mathcal{L}(\mathbf{w})\| \leq B$. Moreover, let the initial estimate \mathbf{w}_0 satisfy $\|\mathbf{w}_0 - \mathbf{w}^*\| \leq R$. Then,*

by setting the learning rate as $\eta = R/(B \cdot \sqrt{T})$ in Scheme 2.3, when D iterations of LDPC decoding are employed during each gradient descent step, ensures the following:

$$\mathbb{E}[\mathcal{L}(\bar{\mathbf{w}}_T)] - \mathcal{L}(\mathbf{w}^*) \leq RB/((1 - q_D) \cdot \sqrt{T}), \quad (2.20)$$

where $\bar{\mathbf{w}}_T = \frac{1}{T} \cdot \sum_{t \in [T]} \mathbf{w}_t$ and the expectation is taken over the distribution of the stragglers.

Proof. Proof of Theorem 2.6: It follows from the convexity of the loss function $\mathcal{L}(\cdot)$ that

$$\mathcal{L}(\bar{\mathbf{w}}_T) - \mathcal{L}(\mathbf{w}^*) \leq \frac{1}{T} \sum_{t=1}^T \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}^*) \leq \frac{1}{T} \sum_{t=1}^T \nabla \mathcal{L}(\mathbf{w}_t) \cdot (\mathbf{w}_t - \mathbf{w}^*). \quad (2.21)$$

Recall from (2.16) that, for $0 \leq t \leq T - 1$, we have

$$\mathbf{w}_{t+1} = P_{\mathbf{w}}(\mathbf{w}_t - g_t(\mathbf{w}_t)),$$

where $g_t(\mathbf{w}_t) = (\hat{c}(t+1; D)_1, \dots, \hat{c}(t+1; D)_k)^T - \hat{\mathbf{b}}_{t+1}$. Now, consider

$$\begin{aligned} \|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2 &\leq \|P_{\mathbf{w}}(\mathbf{w}_t - \mathbf{g}_t(\mathbf{w}_t)) - \mathbf{w}^*\|^2 \stackrel{(i)}{=} \|P_{\mathbf{w}}(\mathbf{w}_t - \mathbf{g}_t(\mathbf{w}_t)) - P_{\mathbf{w}}(\mathbf{w}^*)\|^2 \\ &\leq \|\mathbf{w}_t - \mathbf{g}_t(\mathbf{w}_t) - \mathbf{w}^*\|^2 \\ &= \|\mathbf{w}_t - \mathbf{w}^*\|^2 - 2\eta \cdot \langle \mathbf{g}_t(\mathbf{w}_t), (\mathbf{w}_t - \mathbf{w}^*) \rangle + \eta^2 \|\mathbf{g}_t(\mathbf{w}_t)\|^2 \\ &\leq \|\mathbf{w}_t - \mathbf{w}^*\|^2 - 2\eta \cdot \langle \mathbf{g}_t(\mathbf{w}_t), (\mathbf{w}_t - \mathbf{w}^*) \rangle + \eta^2 B^2, \end{aligned} \quad (2.22)$$

where (i) follows from the fact that $\mathbf{w}^* \in \mathcal{W}$ and (ii) holds as the operator $P_{\mathbf{w}}$ is non-expanding, i.e.,

$$\|P_{\mathbf{w}}(\mathbf{w}) - P_{\mathbf{w}}(\mathbf{w}')\| \leq \|\mathbf{w} - \mathbf{w}'\| \quad \text{for all } \mathbf{w}, \mathbf{w}' \in \mathbb{R}^d.$$

Let \mathcal{H}_t denote the history, i.e., identity of the stragglers, before the $(t + 1)$ -th step of the optimization procedure. Note that it follows from Lemma 2.5 that

$$\mathbb{E}[\mathbf{g}_t(\mathbf{w}_t) \mid \mathcal{H}_t] = (1 - q_D) \cdot \nabla \mathcal{L}(\mathbf{w}_t). \quad (2.23)$$

By combining (2.22) and (2.23), we obtain that

$$\mathbb{E}[\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2 \mid \mathcal{H}_t] \leq \|\mathbf{w}_t - \mathbf{w}^*\|^2 - 2\eta \cdot (1 - q_D) \cdot \langle \nabla \mathcal{L}(\mathbf{w}_t), (\mathbf{w}_t - \mathbf{w}^*) \rangle + \eta^2 B^2. \quad (2.24)$$

Now taking expectation on the both sides gives us that

$$\mathbb{E}[\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2] \leq \mathbb{E}[\|\mathbf{w}_t - \mathbf{w}^*\|^2] - 2 \cdot \mathbb{E}[\eta \cdot (1 - q_D) \cdot \langle \nabla \mathcal{L}(\mathbf{w}_t), (\mathbf{w}_t - \mathbf{w}^*) \rangle] + \eta^2 B^2. \quad (2.25)$$

or

$$(1 - q_D) \cdot \mathbb{E}[\langle \nabla \mathcal{L}(\mathbf{w}_t), (\mathbf{w}_t - \mathbf{w}^*) \rangle] \leq \frac{1}{2\eta} \cdot \mathbb{E}[\|\mathbf{w}_t - \mathbf{w}^*\|^2] - \frac{1}{2\eta} \cdot \mathbb{E}[\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2] + \frac{\eta B^2}{2}. \quad (2.26)$$

By taking the average of the aforementioned inequality over T iteration, we obtain that

$$\begin{aligned} \mathbb{E} \left[\frac{1}{T} \sum_{t=0}^{T-1} \langle \nabla \mathcal{L}(\mathbf{w}_t), (\mathbf{w}_t - \mathbf{w}^*) \rangle \right] &\leq \frac{1}{2\eta(1 - q_D)} \cdot \left(\frac{\mathbb{E}[\|\mathbf{w}_0 - \mathbf{w}^*\|^2]}{T} - \frac{\mathbb{E}[\|\mathbf{w}_T - \mathbf{w}^*\|^2]}{T} + \eta^2 B^2 \right) \\ &\leq \frac{\|\mathbf{w}_0 - \mathbf{w}^*\|^2}{2\eta T(1 - q_D)} + \frac{\eta B^2}{2(1 - q_D)} \\ &\leq \frac{R^2}{2\eta T(1 - q_D)} + \frac{\eta B^2}{2(1 - q_D)} \\ &\stackrel{(i)}{\leq} \frac{1}{1 - q_D} \cdot \frac{RB}{\sqrt{T}}, \end{aligned} \quad (2.27)$$

where (i) follows from the choice of η . Now, Theorem 2.6 follows by combining (2.21) and (2.27). \square

2.3 Simulation Results

In this section, we conduct a detailed evaluation of our moment encoding based scheme (cf. Scheme 2.3) for distributed computation. In particular, we perform experiments on distributed setting to obtain solutions of two problems: 1) Least-square estimation, and 2) Sparse recovery. Recall that, for least-square estimation, given inputs $\mathbf{y} \in \mathbb{R}^n$ and $\mathbf{X} \equiv \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\} \in \mathbb{R}^{n \times d}$ the task is to find $\arg \min_{\mathbf{w} \in \mathbb{R}^d} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$. Note that this problem does not require a projection step during the optimization procedure. In the sparse recovery problem, one seeks to find a u -sparse vector $\mathbf{w} \in \mathbb{R}^d$ (this means at most u coordinates out of d of the vector \mathbf{w} are nonzero) from linear samples $\mathbf{y} = \mathbf{X}\mathbf{w}$, for some matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$. In this case, t -th step of the projected gradient descent procedure takes the form [50] $\mathbf{w}_t = H_u(\mathbf{w}_{t-1} - \eta \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}_{t-1}))$, where $\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \mathbf{X}^T \mathbf{X} \mathbf{w} - \mathbf{X}^T \mathbf{y}$ is the gradient of the squared loss $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$ and $H_u(\mathbf{w})$ is the thresholding operation that sets all except the largest u coordinates in absolute value of $\mathbf{w} \in \mathbb{R}^d$ to zero. To compute the gradient we again employ the moment encoding method with LDPC codes as outlined in Scheme 2.3. Note that the thresholding operation can be easily performed by the master node itself.

Figure 2.1 presents the results for the least-square estimation problem. In our experiments, the data samples $\mathbf{X} \equiv \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\} \subset \mathbb{R}^d$ are randomly generated with the dimensions $d \in \{200, 400, 800, 1000\}$ and the number of total samples $n = 2048$. The corresponding labels \mathbf{y} are created by multiplying the data matrix \mathbf{X} with randomly drawn vector $\mathbf{w}^* \in \mathbb{R}^d$. We implement Scheme 2.3 on a real-life distributed computing framework (swarm2) at the University of Massachusetts Amherst [119] using mpi4py Python package. The setup involves a cluster of 41 computing nodes (40 worker nodes and 1 master nodes). Throughout this section, the plotted results are averaged over 100 trials. We compare our LDPC codes based (rate= 1/2) moment encoding scheme with the recently proposed data encoding (with MDS/Gaussian

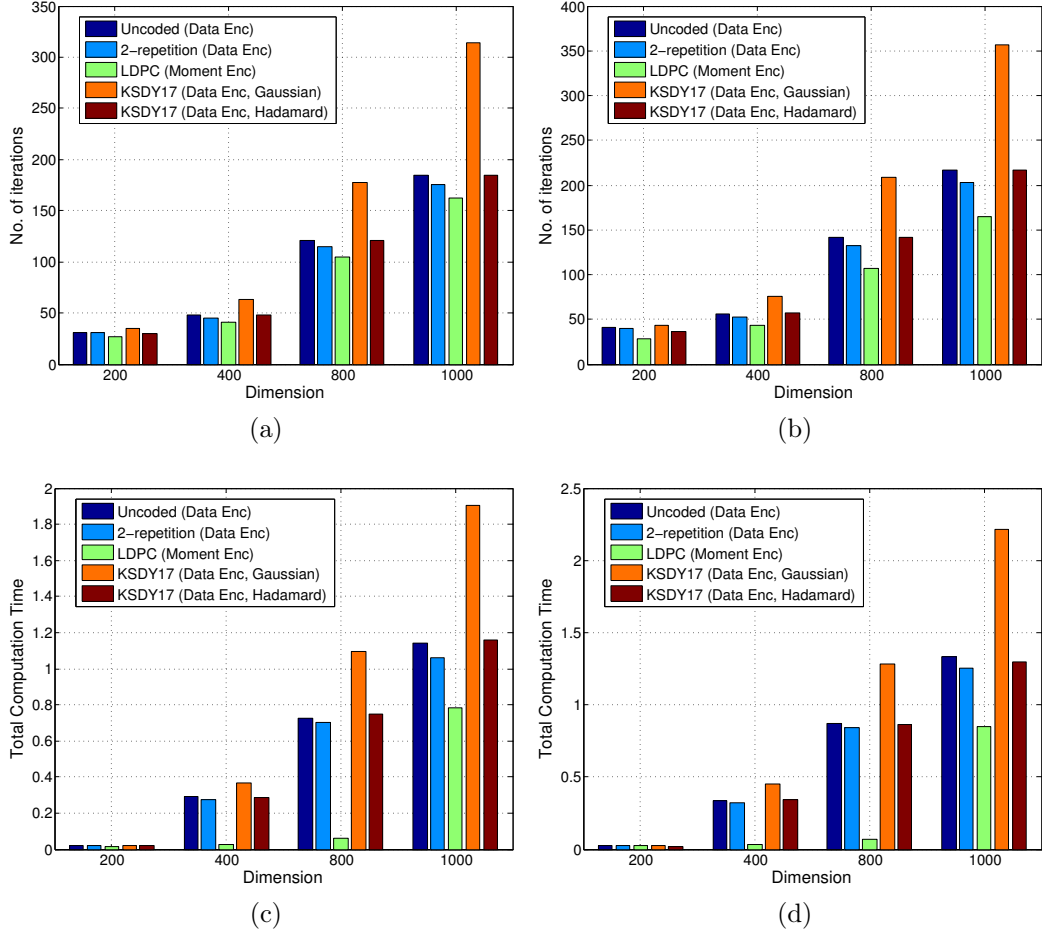


Figure 2.1: Total number of iterations and total computation time for solving the linear regression problem ($m = 2048$). The number of stragglers are 5 (top-left), 10 (top-right), 5 (bottom-left) and 10 (bottom-right).

matrices) scheme of Karakus et al. (KSDY17 in the figures) [67], as well as with uncoded and replication-based schemes (2-replication).⁶ In all cases, we wait for either 30 or 35 workers to respond before the computations at the master node, i.e., the number of stragglers is 10 or 5, respectively. In order to implement our scheme, we utilize a $(40, 20)$ LDPC code. In the replication-based schemes, we partition the data and repeat each partition of the data twice. We use sub-sampled Hadamard and

⁶Here, we do not compare our scheme with the approaches proposed in [120] and [84] as both of these schemes involve significantly different computation and communication requirements. For example, the gradient coding scheme [120] requires communicating d -dimensional vectors; and the approach of [84] involves encoding of two different matrices and two rounds of communications per step of the optimization procedure.

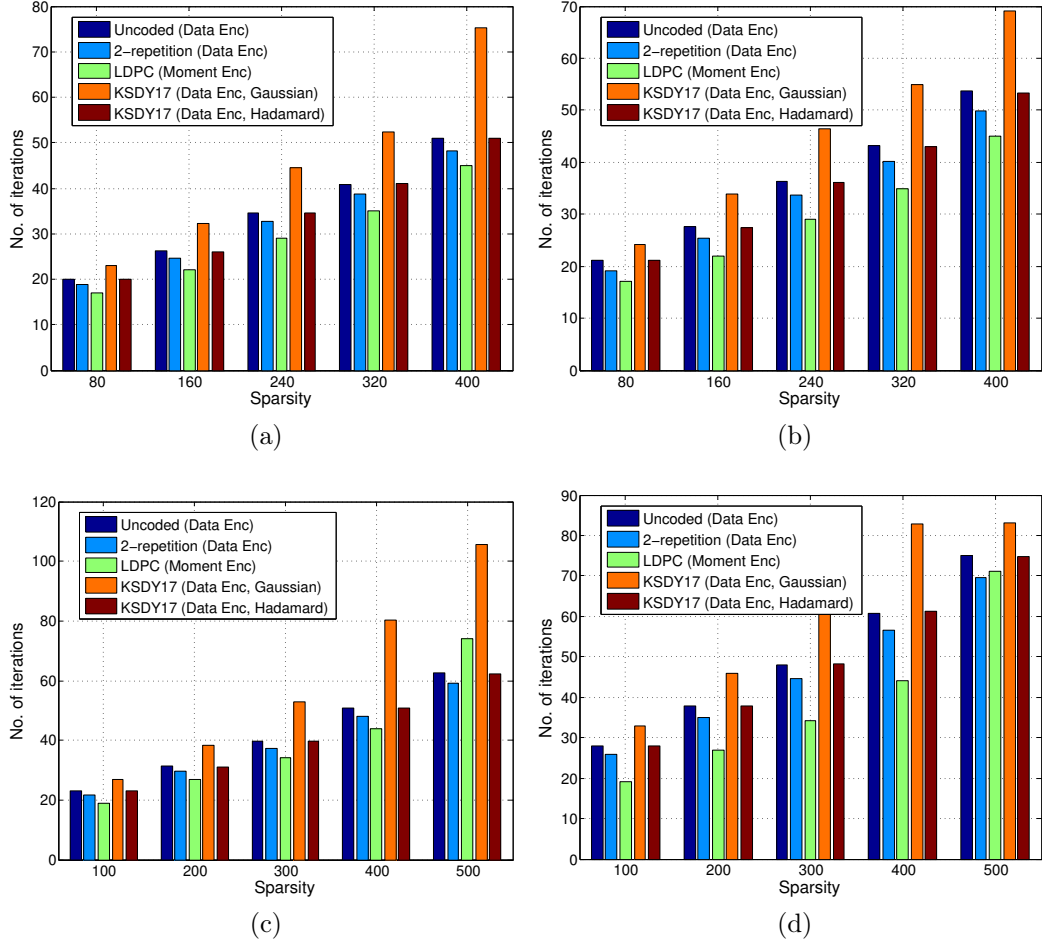


Figure 2.2: Total number of iterations for solving sparse recovery problem in an overdetermined system ($m = 2048$). The left two figures correspond to the dimension 800 and the remaining ones correspond to dimension 1000. The number of stragglers are 5(top-left), 10 (top-right), 5 (bottom-left) and 10 (bottom-right).

Gaussian matrices to implement the data encoding method from [67]. We sampled the columns of 4096×4096 Hadamard matrix and generated 4096×2048 random Gaussian matrices for the purpose of our experiments. For each case we record the number of steps until the Euclidean distance of the evaluated parameter from the actual parameter vector \mathbf{w}^* is within a small threshold.

For the sparse recovery problem, we consider both the over-determined ($n > d$) and the under-determined ($n < d$) cases. For $n > d$, we adopt the same experimental setup as described above with $n = 2048$, but restrict ourselves to the dimensions $d \in \{800, 1000\}$. For each d , we consider different sparsities: for $f \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$,

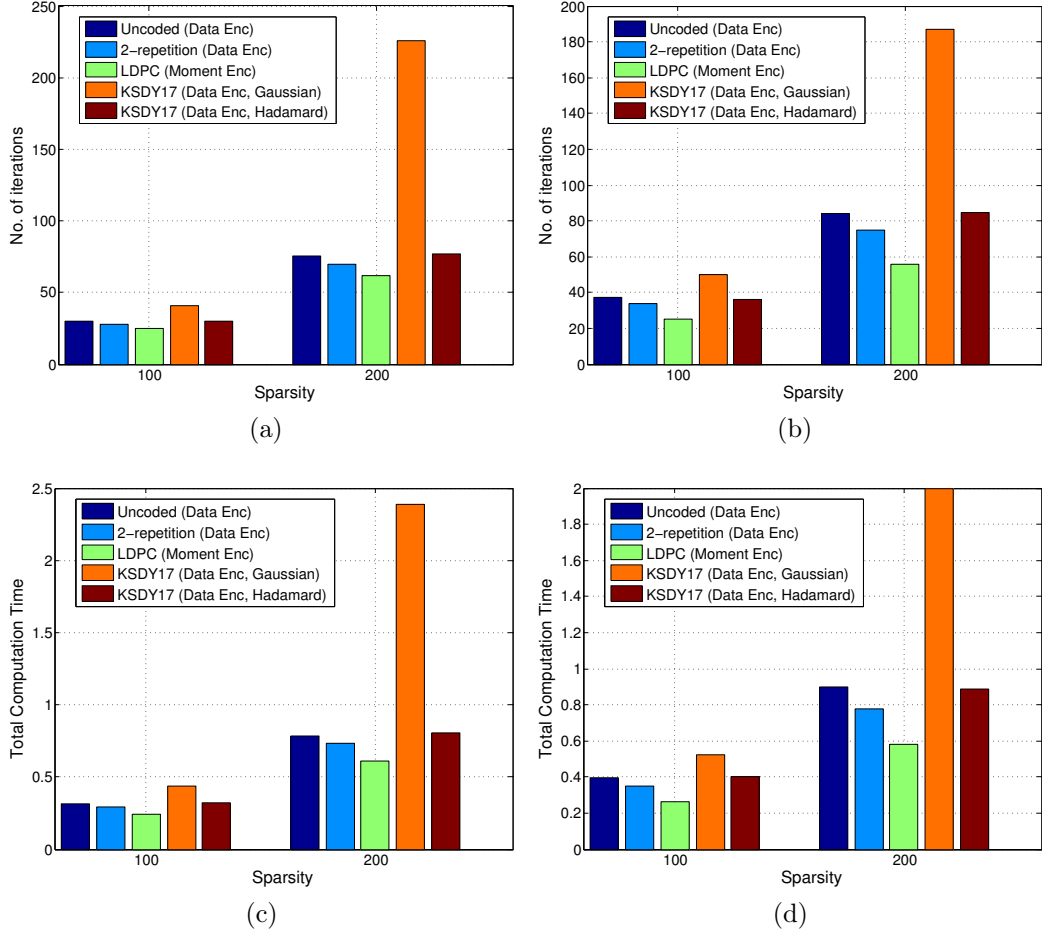


Figure 2.3: Number of iterations and computation time for the sparse recovery problem in an underdetermined system ($k = 2000, m = 1024$). The number of stragglers are 5 (top-left), 10 (top-right), 5 (bottom-left) and 10 (bottom-right).

$u = d \cdot f$ entries in \mathbf{w}^* are nonzero. Figure 2.2 presents the results for the sparse recovery problem in this over-determined setup. We only plot the number of steps of the optimization procedure. The total computation time shows a similar trend. For $n < d$, we generate the matrix \mathbf{X} as a 1024×2000 matrix with i.i.d. entries distributed according to the standard normal distribution. The true parameter vector \mathbf{w}^* is drawn randomly with sparsity levels $u \in \{100, 200\}$. The results obtained from our experiments are presented in Figure 2.3. As it is evident from the plots in Figure 2.1, 2.2 and 2.3, our scheme requires smaller number of steps to converge to the true model parameters. Furthermore, our scheme also leads to smaller overall computation time.

2.4 Conclusion and Future Direction

In this chapter, we propose to encode the second moment of the data for the purpose of mitigating the effect of the stragglers in the distributed gradient descent algorithm. This scheme is specifically tailored for the purpose of squared loss function as the idea revolves around the the second moment of the loss function. This idea can be easily generalized to other loss functions -such as logarithmic loss or the Poisson loss function. We used LDPC code for the purpose of encoding and took advantage of the decoding scheme to tune the algorithm ensuring the desired quality of the estimate of the gradient.

One very serious direction of future work is to figure out encoding scheme for general loss function especially non-linear loss function that appears in training neural network (relu network). It would be interesting to see what functional of the data that needs to be encoded for the purpose of the straggler mitigation. One important thing is that all the encoding scheme including ours work on the linear operation specifically the Matrix-vector product part of the computations. But encoding for non-linear computation is still a open problem. Also in this chapter, we show the convergence analysis for the case of the randomized straggler case where each machine can be straggler with some probability. Even though the scheme will work for the adversarial case, further analysis is required.

CHAPTER 3

VQSGD: VECTOR QUANTIZED STOCHASTIC GRADIENT DESCENT

In the previous chapter, we focus on the issue of *stragglers* in the distributed SGD. Now we focus on the issue of high communication overhead between parameter (central) server and worker nodes which can cause *bottleneck* to the efficiency and speed of learning using SGD [28]. In the recent years, various quantization and sparsification techniques [3, 6, 12, 74, 90, 107, 118, 125, 131] have been developed to alleviate the problem of communication bottleneck. In this chapter, we present a family of *privacy-preserving vector-quantization* schemes that incur low communication costs while providing convergence guarantees. In particular, we provide explicit and efficient quantization schemes based on convex hull of specific structured point sets in \mathbb{R}^d that require $O(d \log d/R^2)$ bits to communicate an unbiased gradient estimate that has variance bounded above by R^2 : this is within a $\log d$ factor of the optimal amount of communication that is necessary and sufficient for this purpose.

At a high level, our scheme is based on the idea that any vector $\mathbf{v} \in \mathbb{R}^d$ with bounded norm can be represented as a convex combination of a carefully constructed point set $C \subset \mathbb{R}^d$. This convex combination essentially allows us to choose a point $\mathbf{c} \in C$ with probability proportional to its coefficient, which makes it an unbiased estimator of \mathbf{v} . The bound on the variance is obtained from the circumradius of the convex hull of C . Moreover, communicating the unbiased estimate is equivalent to communicating the index of $\mathbf{c} \in C$ (according to some fixed ordering) that requires only $\log |C|$ bits.

Large convex hulls have small variation in the coefficients of the convex combination of any two points of bounded norm. This observation allows us to obtain ϵ -differential privacy (for any $\epsilon > \epsilon_0$), where ϵ_0 depends on the choice of the point set. We also propose Randomized Response (RR) [130] and RAPPOR [45] based mechanisms that can be used over the proposed quantization to achieve ϵ -differential privacy (for any $\epsilon > 0$) with small trade-off in the variance of the estimates.

The family of schemes described above is fairly general and can be instantiated using different structured point sets. The cardinality of the point set bounds the communication cost of the quantization scheme. Whereas, the diameter of the point set dictates the variance bounds and the privacy guarantees of the scheme.

We provide a strong characterization of the point-sets that can be used for our quantization scheme. Using this characterization, we propose construction of point-sets that allow us to attain a smooth trade-off between variance and communication of the quantization scheme. We also propose some explicit structured point sets and show tradeoff in the various parameters guaranteed by them. Our results¹ (summarized in Table 3.1) are the first quantization schemes in literature to achieve privacy directly through quantization. While our randomized construction is optimal in terms of communication, the explicit schemes are within $\log d$ factor of a lower bound that we provide.

Empirically we compare our quantization schemes to the state-of-art schemes [6, 118]. We observe that our cross-polytope vqSGD, performs equally well in practice, while providing asymptotic reduction in the communication cost. The communication results are compared in Table 3.2(up).

While differential privacy for gradient based algorithms [2, 110] were considered earlier in literature, cpSGD [5] is the only work that considered achieving differential

¹Note that ϵ denotes the privacy parameter and ε refers to the packing parameter of ε -nets.

² O_ε hides terms involving ε

Point set	Error	Communication (bits)	Privacy	Efficiency
Gaussian-Sampling (Theorem 3.7) for any $c > \log(d)$	$\frac{d}{cN}$	Nc	-	$O(\exp(c))$
Reed-Muller (C_{RM}) (Proposition 3.9)	$\frac{d}{N}$	$N \log 2d$	-	$O(d)$
Cross-polytope (C_{cp}) (Proposition 3.10)	$\frac{d}{N}$	$N \log 2d$	$\epsilon > O(\log d)$	$O(d)$
Scaled ε -Net (C_{net}) (Proposition 3.12)	$\frac{1}{N}$	$O_\varepsilon(Nd)^2$	-	$O\left(\left(\frac{1}{\varepsilon}\right)^d\right)$
Simplex (C_S) (Proposition 3.13)	$\frac{d^2}{N}$	$N \log(d+1)$	$\epsilon > \log 7$	$O(d)$
Hadamard (C_H) (Proposition 3.14)	$\frac{d^2}{N}$	$N \log d$	$\epsilon > \log(1 + \sqrt{2})$	$O(d)$
Cross-polytope (C_{cp}) + RR (Theorem 3.16)	$\frac{d^2}{N}$	$N \log(2d)$	$\epsilon > 0$	$O(d)$
Cross-polytope (C_{cp}) + RAPPOR (Theorem 3.17)	$\frac{d^2}{N}$	$2Nd$	$\epsilon > 0$	$O(d)$

Table 3.1: List of results. (N : number of worker nodes, d : dimension).

privacy for gradient based algorithms and simultaneously minimizing the gradient communication cost. The authors propose a binomial mechanism to add discrete noise to the quantized gradients to achieve communication-efficient (ϵ, δ) -differentially private gradient descent with convergence guarantees. The quantization schemes used are similar to those presented in [118] and hence require $\Omega(d)$ bits of communication per compute node. The parameters of the binomial noise are dictated by the required privacy guarantees which in turn controls the communication cost.

In this work we show that certain instantiations of our quantization schemes are ϵ -differentially private. Note that this is a much stronger privacy notion than (ϵ, δ) -privacy. Moreover, we get this privacy guarantee directly from the quantization schemes and hence the communication cost remains sublinear ($\log d$) in dimension. We also propose a Randomized Response [130] based private-quantization scheme that requires $O(\log d)$ bits of communication per compute node to get an ϵ -differential privacy while losing a factor of $O(d)$ in convergence rate. Table 3.2(down) compares

the guarantees provided by our private quantization schemes with the results of cpSGD [5].

Organization: In Section 3.1, we describe some other related work on communication efficiency in the federated learning setup. We start in Section 3.2 describing the settings for our results. The vqSGD quantization scheme is presented in Section 3.3. In Section 3.4 we provide a handle to test whether a point-set is a valid vqSGD scheme, and prove existence of a point-set that achieves a communication cost equal to the dimension divided by the variance, which matches a lower bound we prove. We provide a few structured deterministic constructions of point sets in this section as well. Section 3.5 emphasizes the privacy component of vqSGD - and derives the privacy parameters of several vqSGD schemes. Finally, we provide some experiments to support vqSGD in Section 3.6.

Method	Error	Comm
QSGD [6]	$\min\{\frac{d}{s^2}, \frac{\sqrt{d}}{s}\} \frac{1}{N}$	$Ns(s + \sqrt{d})$
DME [118]	$\min\{\frac{1}{Ns}, \frac{\log d}{N(s-1)^2}\}$	Nsd
vqSGD $Q_{C_{cp}}$	$\frac{d}{Ns}$	$Ns \log d$
Gaussian	$\frac{d}{Nsc}$	Nsc

Method	Error	Comm	DP (ϵ)
cpSGD [5]	$O_\delta \left(\frac{d}{N}\right)^3$	$O_\delta(d)$	$\delta > 0,$ $\epsilon > f(\delta)$
vqSGD $Q_{C_{cp}}$	$O\left(\frac{d}{N}\right)$	$O(\log d)$	$\epsilon > O(\log d)$
vqSGD Q_{C_S}	$O\left(\frac{d^2}{N}\right)$	$O(\log d)$	$\epsilon > \log 7$
vqSGD Q_{C_H}	$O\left(\frac{d^2}{N}\right)$	$O(\log d)$	$\epsilon > \log(2.5)$
vqSGD $Q_{C_{cp}}$ + RR	$O\left(\frac{d^2}{N}\right)$	$O(\log d)$	$\epsilon > 0$

Table 3.2: (Up): Comparison of non private quantization schemes. (Down): Comparison of private quantization schemes. (N : number of worker nodes, s, c : tuning parameter (≥ 1))

3.1 Related Work

The foundations of gradient quantization was laid by [103] and [114] with schemes that require the compute nodes to send exactly 1-bit per coordinate of the gradient. They also suggested using local error accumulation to correct the global gradient in every iteration. While these novel techniques worked well in practice, there were no theoretical guarantees provided for convergence of the scheme. These seminal works fueled multiple research directions.

Quantization & Sparsification: [6, 125, 131] propose stochastic quantization techniques to represent each coordinate of the gradient using small number of bits. The proposed schemes always return an unbiased estimator of the local gradient and require $c = \Omega(\sqrt{d})$ bits of communication to compute the global gradient with variance bounded by a multiplicative factor of $O(d/c)$. The quantization techniques for distributed SGD, can be used in the more general setting of communication efficient distributed mean estimation problem, which was the focus of [118]. The quantization schemes proposed in [118] require $O(d)$ bits of communication per compute node to estimate the global mean with a constant (independent of d) squared error (variance). Even though the tradeoff between communication and accuracy achieved by the above mentioned schemes are near optimal [139], they were unable to break the \sqrt{d} barrier of communication cost. Moreover, the schemes proposed in [6, 118] are variable length codes that achieve low communication in expectation. The worst case communication cost could be higher. In a parallel work [90], the authors propose an efficient fixed-length quantization scheme that achieves near-optimal convergence with T -rounds of SGD. However, the goal of their work is different from ours, and the methodologies are different as well.

³ O_δ hides terms involving δ .

In this work, we propose (fixed length) quantization schemes that require $o(d)$ (as low as $\log d$) bits of communication and are almost optimal as well. In fact for any c -bits of communication, the quantization scheme with Gaussian points achieves a variance of $O(d/c)$ that meets the lower bounds for any unbiased quantization scheme (also shown in the current work).

Gradient sparsification techniques with provable convergence (under standard assumptions) were studied in [3, 8, 62, 113]. The main idea in these techniques is to communicate only the top- k *components* of the d -dimensional local gradients that can be accumulated globally to obtain a good estimate of the true gradient. Unlike the quantization schemes described above, gradient sparsification techniques can achieve $O(\log d)$ bits of communication, but are not usually unbiased estimates of the true gradients. [107] suggest randomized sparsification schemes that are unbiased, but are not known to provide any theoretical convergence guarantees in very low sparsity regimes.

See Table 3.2 for a comparison of our results with the state of the art quantization schemes.

Error Feedback: Many works focused on providing techniques to reduce the error incurred due to quantization [61, 70] using locally accumulated errors. In this work, we focus primarily on gradient quantization techniques, and note that the variance reduction techniques of [61] can be used on top of the proposed quantization schemes.

3.2 Preliminaries

For any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, we denote the Euclidean (ℓ_2) distance between them as $\|\mathbf{x} - \mathbf{y}\|_2$. For any vector $\mathbf{x} \in \mathbb{R}^d$, x_i denotes its i -th coordinate. For any $\mathbf{c} \in \mathbb{R}^d$, and $r > 0$, let $B_d(\mathbf{c}, r)$ denote a d -dimensional ℓ_2 ball of radius r centered at \mathbf{c} . Let $\mathbf{e}_i \in \mathbb{R}^d$ denote

the i -th standard basis vector which has 1 in the i -th position and 0 everywhere else. Also, for any prime power q , let \mathbb{F}_q denote a finite field with q elements.

For a discrete set of points $C \subset \mathbb{R}^d$, let $\text{CONV}(C)$ denote the convex hull of points in C , *i.e.*, $\text{CONV}(C) := \{\sum_{\mathbf{c} \in C} a_{\mathbf{c}} \mathbf{c} \mid a_{\mathbf{c}} \geq 0, \sum_{\mathbf{c} \in C} a_{\mathbf{c}} = 1\}$.

Suppose $\mathbf{w} \in \mathbb{R}^d$ be the parameters of a function to be learned (such as weights of a neural network). In each step of the SGD algorithm, the parameters are updated as $\mathbf{w} \leftarrow \mathbf{w} - \eta \hat{\mathbf{g}}$, where η is a possibly time-varying learning rate and $\hat{\mathbf{g}}$ is a stochastic unbiased estimate of \mathbf{g} , the true gradient of some loss function with respect to \mathbf{w} . The assumption of unbiasedness is crucial here, that implies $\mathbb{E} \hat{\mathbf{g}} = \mathbf{g}$.

The goal of any gradient quantization scheme is to reduce cost of communicating the gradient, *i.e.*, to act as an first-order oracle, while not compromising too much on the *quality* of the gradient estimate. The quality of the gradient estimate is measured in terms the convergence guarantees it provides. In this work, we develop a scheme that is an *almost surely bounded oracle* for gradients, *i.e.*, $\|\hat{\mathbf{g}}\|_2^2 \leq B$ with probability 1, for some $B > 0$. The convergence rate of the SGD algorithm for any convex function f depends on the upper bound of the norm of the unbiased estimate, *i.e.*, B , cf. any standard textbook such as [106].

Although we provide an almost surely bounded oracle as our quantization scheme, previous quantization schemes, such as [6], provides a *mean square bounded oracle*, *i.e.*, an unbiased estimate $\hat{\mathbf{g}}$ of \mathbf{g} such that $\mathbb{E} \|\hat{\mathbf{g}}\|_2^2 \leq B$ for some $B > 0$. It is known that, even with a mean square bounded oracle, SGD algorithm for a convex function converges with dependence on the upper bound B [17]. As discussed in [6], one can also consider the variance of $\hat{\mathbf{g}}$ without any palpable difference in theory or practice. Therefore, below we consider the variance of the estimate $\hat{\mathbf{g}}$ as the main measure of error.

In distributed setting with N worker nodes, let \mathbf{g}_i and $\hat{\mathbf{g}}_i$ are the local true gradient and its unbiased estimate computed at the i th compute node for some $i \in \{1, \dots, N\}$.

For $\mathbf{g} = \frac{1}{N} \sum_i \mathbf{g}_i$, the variance of the estimate $\hat{\mathbf{g}} = \frac{1}{N} \sum_i \hat{\mathbf{g}}_i$ is defined as

$$\text{Var}(\hat{\mathbf{g}}) := \mathbb{E} \left[\left\| \frac{1}{N} \sum_{i=1}^N \mathbf{g}_i - \frac{1}{N} \sum_{i=1}^N \hat{\mathbf{g}}_i \right\|_2^2 \right] = \frac{1}{N^2} \sum_{i=1}^N \mathbb{E} [\|\mathbf{g}_i - \hat{\mathbf{g}}_i\|_2^2].$$

In this work, our goal is to design quantization schemes to efficiently compute unbiased estimate $\hat{\mathbf{g}}_i$ of \mathbf{g}_i such that $\text{Var}(\hat{\mathbf{g}})$ is minimized.

For the privacy preserving gradient quantization schemes, we consider the standard notion of (ϵ, δ) -differential privacy (DP) as defined in [44]. Consider data-sets from a domain \mathcal{X} . Two data-sets $U, V \in \mathcal{X}$, are neighboring if they differ in at most one data point.

Definition 3.1. *A randomized algorithm \mathcal{M} with domain \mathcal{X} is (ϵ, δ) -differentially private (DP) if for all $S \subset \text{Range}(\mathcal{M})$ and for all neighboring data sets $U, V \in \mathcal{X}$,*

$$\Pr[\mathcal{M}(U) \in S] \leq e^\epsilon \Pr[\mathcal{M}(V) \in S] + \delta,$$

where, the probability is over the randomness in \mathcal{M} . If $\delta = 0$, we say that \mathcal{M} is ϵ -DP.

We will need the notion of an ϵ -nets subsequently.

Definition 3.2 (ϵ -net). *A set of points $N(\epsilon) \subset S^{d-1}$ is an ϵ -net for the unit sphere S^{d-1} if for any point $\mathbf{x} \in S^{d-1}$ there exists a net point $\mathbf{u} \in N(\epsilon)$ such that $\|\mathbf{x} - \mathbf{u}\|_2 \leq \epsilon$.*

There exist various constructions for ϵ -net over the unit sphere in \mathbb{R}^d of size at most $(1 + 2/\epsilon)^d$ [30].

3.3 Quantization Scheme

We first present our quantization scheme in full generality. Individual quantization schemes with different tradeoffs are then obtained as specific instances of this general scheme.

Let $C = \{\mathbf{c}_1, \dots, \mathbf{c}_m\} \subset \mathbb{R}^d$ be a discrete set of points such that its convex hull, $\text{CONV}(C)$ satisfies

$$B_d(\mathbf{0}_d, 1) \subset \text{CONV}(C) \subseteq B_d(\mathbf{0}_d, R), R > 1. \quad (3.1)$$

Let $\mathbf{v} \in B_d(\mathbf{0}_d, 1)$. Since $B_d(\mathbf{0}_d, 1) \subseteq \text{CONV}(C)$, we can write \mathbf{v} as a convex linear combination of points in C . Let $\mathbf{v} = \sum_{i=1}^m a_i \mathbf{c}_i$, where $a_i \geq 0, \sum_{i=1}^m a_i = 1$. We can view the coefficients of the convex combination (a_1, \dots, a_m) as a probability distribution over points in C . Define the quantization of \mathbf{v} with respect to the set of points C as follows:

$$Q_C(\mathbf{v}) := \mathbf{c}_i \text{ with probability } a_i$$

It follows from the definition of the quantization that $Q_C(\mathbf{v})$ is an unbiased estimator of \mathbf{v} .

Lemma 3.1. $\mathbb{E}(Q_C(\mathbf{v})) = \mathbf{v}$.

Proof of Lemma 3.1. $\mathbb{E}[Q_C(\mathbf{v})] = \sum_{i=1}^{|C|} a_i \cdot \mathbf{c}_i = \mathbf{v}$. □

We assume that C is fixed in advance and is known to the compute nodes and the parameter server.

Remark 3.1. *Communicating the quantization of any vector \mathbf{v} , amounts to sending a floating point number $\|\mathbf{v}\|_2$, and the index of point $Q_C(\mathbf{v})$ which requires $\log |C|$ bits. For many loss functions, such as Lipschitz functions, the bound on the norm of the gradients is known to both the compute nodes and the parameter server. Therefore, we can avoid sending $\|\mathbf{v}\|_2$ and the cost of communicating the gradients is then exactly $\log |C|$ bits.*

Remark 3.2. *If, for any vector \mathbf{v} , we send the floating point number $\|\mathbf{v}\|_2$ separately, instead of there being a known upper bound on gradient, we can just assume*

without loss of generality that $\mathbf{v} \in S^{d-1}$. In this case, in the subsequent bounds on variance in this section, R^2 can be replaced by $R^2 - 1$

Any point set C that satisfies Condition (3.1) gives the following bound on the variance of the quantizer.

Lemma 3.2. *Let $C \subset \mathbb{R}^d$ be a point set satisfying Condition (3.1). For any $\mathbf{v} \in B_d(\mathbf{0}_d, 1)$, let $\hat{\mathbf{v}} := Q_C(\mathbf{v})$. Then, $\|\hat{\mathbf{v}}\|_2^2 \leq R^2$ almost surely, and $\mathbb{E}[\|\mathbf{v} - \hat{\mathbf{v}}\|_2^2] \leq R^2$.*

Proof of Lemma 3.2. From the definition of the quantization function,

$$\mathbb{E}[\|\mathbf{v} - Q_C(\mathbf{v})\|_2^2] = \mathbb{E}[\|Q_C(\mathbf{v})\|_2^2] - \|\mathbf{v}\|^2 \leq R^2.$$

This is true as C satisfies Condition (3.1) and therefore, each point $\mathbf{c}_i \in C$ has a bounded norm, $\|\mathbf{c}_i\| \leq R$. \square

From the above mentioned properties, we get a family of quantization schemes depending on the choice of point set C that satisfy Condition (3.1). For any choice of quantization scheme from this family, we get the following bound regarding the convergence of the distributed SGD.

Theorem 3.3. *Let $C \subset \mathbb{R}^d$ be a point set satisfying Condition (3.1). Let $\mathbf{g}_i \in \mathbb{R}^d$ be the local gradient computed at the i -th node, Define $\hat{\mathbf{g}} := \frac{1}{N} \sum_{i=1}^N \hat{\mathbf{g}}_i$, where $\hat{\mathbf{g}}_i := \|\mathbf{g}_i\| \cdot Q_C(\mathbf{g}_i/\|\mathbf{g}_i\|)$. Then,*

$$\mathbb{E}[\hat{\mathbf{g}}] = \mathbf{g} \quad \text{and} \quad \mathbb{E}[\|\mathbf{g} - \hat{\mathbf{g}}\|_2^2] \leq (R/N)^2 \sum_i \|\mathbf{g}_i\|^2.$$

Proof of Theorem 3.3. Since $\hat{\mathbf{g}}$ is the average of N unbiased estimators, the fact that $\mathbb{E}[\hat{\mathbf{g}}] = \mathbf{g}$ follows from Lemma 3.1. For the variance computation, note that

$$\begin{aligned}\mathbb{E}[\|\mathbf{g} - \hat{\mathbf{g}}\|_2^2] &= \frac{1}{N^2} \left(\sum_{i=1}^N \mathbb{E}[\|\mathbf{g}_i - \hat{\mathbf{g}}\|_2^2] \right) \quad (\text{since } \hat{\mathbf{g}} \text{ is an unbiased estimator of } \mathbf{g}) \\ &\leq \frac{R^2}{N^2} \sum_{i=1}^N \|\mathbf{g}_i\|^2 \quad (\text{from Lemma 3.2}).\end{aligned}$$

□

Remark 3.3. *Computing the quantization $Q_C(\cdot)$ amounts to solving a system of $|C|$ linear equations in \mathbb{R}^d . For general point sets C , this takes about $O(|C|^3)$ time (since $|C| \geq d$). However, we show that for certain structured point sets $Q_C(\cdot)$ can be computed in linear time.*

From Theorem 3.3 we observe that the communication cost of the quantization scheme depends on the cardinality of C while the convergence is dictated by the circumradius R of the convex hull of C . In the Section 3.4, we present several constructions of point sets which provide varying tradeoffs between communication and variance of the quantizer.

Reducing Variance: Here, we propose a simple repetition technique to reduce the variance of the quantization scheme. For any $s > 1$, let $Q_C(s, \mathbf{v}) := \frac{1}{s} \sum_{i=1}^s Q_C^{(i)}(\mathbf{v})$ be the average over s independent applications of the quantization $Q_C(\mathbf{v})$. Note that even though $Q_C(s, \mathbf{v})$ is not a point in C , we can communicate $Q_C(s, \mathbf{v})$ using an equivalent representation as a tuple of s independent applications of $Q_C(\mathbf{v})$ that requires $s \log |C|$ bits. Using this repetition technique we see that the variance reduces by factor of s while the communication increases by the exact same factor.

Proposition 3.4. *Let $C \subset \mathbb{R}^d$ be a point set satisfying Condition (3.1). For any $\mathbf{v} \in B_d(\mathbf{0}_d, 1)$, and any $s \geq 1$, let $\hat{\mathbf{v}} := Q_C(s, \mathbf{v})$. Then, $\mathbb{E}[\|\mathbf{v} - \hat{\mathbf{v}}\|_2^2] \leq R^2/s$.*

Proof of Proposition 3.4. The proof follows simply by linearity of expectations and Lemma 3.2.

$$\mathbb{E} [\|\mathbf{v} - \hat{\mathbf{v}}\|_2^2] = \mathbb{E} \left[\left\| \frac{1}{s} \sum_{i=1}^s (\mathbf{v} - Q_C(\mathbf{v})) \right\|^2 \right] \leq \frac{1}{s} \cdot R^2.$$

□

3.4 Constructions of Point Sets and Lower Bound

In this section, we propose constructions of point sets that satisfy Condition (3.1) and provide varying tradeoffs between communication and variance of the quantization scheme. But first, we start with a lower bound that shows that one must communicate $\Omega(\frac{d}{R^2})$ bits to achieve an error of $O(R^2)$ in the estimate of the gradient as per Condition (3.1).

Theorem 3.5. *Let $C \subseteq \mathbb{R}^d$ be a discrete set of points that satisfy Condition (3.1). Then*

$$|C| \geq \exp(\alpha d/R^2)$$

for some absolute constant $\alpha > 0$.

Proof of Theorem 3.5. We use a packing argument for S^{d-1} . Let $\mathbf{c} \in C$. We will estimate the *cardinality* of the set $P(\mathbf{c}) := \{\mathbf{x} \in S^{d-1} : \langle \mathbf{x}, \mathbf{c} \rangle \geq 1\}$ under the uniform measure over S^{d-1} . Using Theorem 3.6, size of C must be at least

$$\frac{\text{area}(S^{d-1})}{\max_{\mathbf{c} \in C} \text{area}(P(\mathbf{c}))}$$

for it to satisfy Condition (3.1).

Note that, $P(\mathbf{c})$ is a hyper-spherical cap with angle ϕ such that $\cos \phi \geq \frac{1}{\|\mathbf{c}\|} \geq \frac{1}{R}$, since C satisfy Condition (3.1). The area of a cap can be computed using the

incomplete beta functions, however a probabilistic argument below will serve to lower bound this.

If we uniformly at random choose a vector \mathbf{z} from S^{d-1} , then the probability p that it is within an angular distance ϕ of a fixed unit vector, \mathbf{u} , will exactly be the ratio of the areas of the hyper-spherical cap and the sphere. Again this probability is known to follow a shifted Beta distribution, but we can estimate it from above using concentration bounds.

Since the area of the hyper-spherical cap is invariant to its center, we can take \mathbf{u} to be the first standard basis vector. It is known that if $\mathbf{g} = (g_1, g_2, \dots, g_d) \in \mathbb{R}^d$ is a random vector with i.i.d. Gaussian $\mathcal{N}(0, 1)$ entries, then $\mathbf{z} := \mathbf{g}/\|\mathbf{g}\|$ is uniform over S^{d-1} . Therefore,

$$\begin{aligned} \Pr(\langle \mathbf{z}, \mathbf{u} \rangle \geq 1/R) &= \Pr(g_1/\|\mathbf{g}\| \geq 1/R) \\ &\leq \Pr(g_1 \geq \|\mathbf{g}\|/R \mid \|\mathbf{g}\| \geq \sqrt{d}/4) + \Pr(\|\mathbf{g}\| < \sqrt{d}/4) \\ &\leq \Pr(g_1 \geq \sqrt{d}/4R) + \Pr(\|\mathbf{g}\| < \sqrt{d}/4). \end{aligned}$$

Now since g_1 is $\mathcal{N}(0, 1)$, $\Pr(g_1 \geq \frac{\sqrt{d}}{4R}) \leq \exp(-\frac{d}{32R^2})$, from Chernoff bound. On the other hand $\|\mathbf{g}\|^2$ is a χ^2 distribution of d degrees of freedom. Since that is sub-exponential, we have $\Pr(\|\mathbf{g}\| < \sqrt{d}/4) \leq \Pr(\|\mathbf{g}\|^2 < d/16) \leq \exp(-\frac{225d}{2048}) \leq \exp(-\frac{d}{32R^2})$ for any $R \geq 1$.

This implies, $|C| \geq (2 \exp(-d/(32R^2)))^{-1}$. \square

We also show a strong characterization of the point sets that satisfy Condition (3.1), and later use this characterization to construct point sets with optimal trade-offs.

Theorem 3.6. *Let $C = \{\mathbf{c}_1, \dots, \mathbf{c}_m\} \subseteq \mathbb{R}^d$ be a discrete set of points. The unit ball $B_d(\mathbf{0}_d, 1) \subseteq \text{CONV}(C)$ if and only if for all points $\mathbf{x} \in S^{d-1}$, there exists a point $\mathbf{c} \in C$ such that $\langle \mathbf{x}, \mathbf{c} \rangle \geq 1$.*

Proof of Theorem 3.6. Assume that for some $\mathbf{x} \in S^{d-1}$, $\langle \mathbf{x}, \mathbf{c} \rangle < 1$ for all $\mathbf{c} \in C$. Which implies that all points of C , and therefore the $\text{CONV}(C)$, are separated from \mathbf{x} by the hyper-plane $H_w := \{\mathbf{w} \in \mathbb{R}^d | \langle \mathbf{x}, \mathbf{w} \rangle = 1\}$. Therefore $\mathbf{x} \notin \text{CONV}(C)$.

To prove the other side, assume $B_d(\mathbf{0}_d, 1) \not\subset \text{CONV}(C)$. Let $H_w := \{\mathbf{z} \in \mathbb{R}^d | \langle \mathbf{w}, \mathbf{z} \rangle = 1\}$ be the separating hyper-plane that partitions $B_d(\mathbf{0}_d, 1)$ such that $\text{CONV}(C)$ lies on one side of the hyper-plane. Without loss of generality, we assume $\text{CONV}(C) \subset H_w^- := \{\mathbf{z} \in \mathbb{R}^d | \langle \mathbf{w}, \mathbf{z} \rangle < 1\}$. Since H_w partitions the unit ball, the distance of H_w from the origin is $1/\|\mathbf{w}\| \leq 1$.

Now consider the point $\mathbf{x} := \mathbf{w}/\|\mathbf{w}\| \in S^{d-1}$. For this point, $\langle \mathbf{x}, \mathbf{c} \rangle = \frac{1}{\|\mathbf{w}\|} \langle \mathbf{w}, \mathbf{c} \rangle < 1$ for all $\mathbf{c} \in C$. \square

3.4.1 Gaussian Point Set

We provide a randomized construction of point set using the characterization defined above, that is optimal in terms of communication.

Theorem 3.7. *Let $R \in [5, 6\sqrt{d}]$. There exists a set C of $\exp(O(d/R^2 + \log d))$ points of ℓ_2 norm at most R each, that satisfy Condition (3.1).*

This theorem is one of our main results and is proved by choosing $\exp(O(d/R^2))$ i.i.d. zero-mean spherical Gaussian vectors with variance $\sigma^2 := R^2/9d$ for each coordinate, and then using them as a covering for an ε -net of the unit sphere with properly chosen ε .

To prove Theorem 3.7, we first show the following lemma that allows us to union bound over the discrete set of points in an ε -net of a unit sphere. Consider an ε -net for the unit sphere $N(\varepsilon)$ for any $\varepsilon \leq 1/R$. We know that such a set exists with $|N(\varepsilon)| \leq (1 + \frac{2}{\varepsilon})^d \leq (\frac{3}{\varepsilon})^d$ [30].

Lemma 3.8. *Let C be a set of points in \mathbb{R}^d such that $\|\mathbf{c}\|^2 \leq R^2$ for all $\mathbf{c} \in C$. If for each $\mathbf{y} \in N(\varepsilon)$, $\mathbf{y}^T \mathbf{c} \geq 2$ for some $\mathbf{c} \in C$, then it holds that for each $\mathbf{x} \in S^{d-1}$, there is a $\mathbf{c}' \in C$ such that $\mathbf{x}^T \mathbf{c}' \geq 1$.*

Proof. Let $\mathbf{y} \in N(\varepsilon)$ be a net-point, and $\mathbf{c} \in C$ be such that $\mathbf{y}^T \mathbf{c} \geq 2$. Note that all points $\mathbf{x} \in S^{d-1}$ in the ε -neighborhood of \mathbf{y} can be written as $\mathbf{x} = \mathbf{y} + \tilde{\mathbf{y}}$, where $\tilde{\mathbf{y}} \in \mathbb{R}^d$ has norm at most ε . Therefore, $\mathbf{x}^T \mathbf{c} = \mathbf{y}^T \mathbf{c} + \tilde{\mathbf{y}}^T \mathbf{c} \geq 2 - \|\tilde{\mathbf{y}}\| \|\mathbf{c}\| \geq 1$. Since $N(\varepsilon)$ covers the entire unit sphere, it follows that for all points \mathbf{x} on the unit sphere, there will be a $\mathbf{c} \in C$ such that $\mathbf{x}^T \mathbf{c} \geq 1$. \square

Proof of Theorem 3.7. Let us choose the random set C of $t := e^{\frac{20d}{R^2} + 2 \log d}$ points in \mathbb{R}^d in the following way: Each coordinate of any $\mathbf{c} \in C$ is chosen independently from a zero-mean Gaussian distribution with variance $\sigma^2 := \frac{R^2}{9d}$.

We say that a vector $\mathbf{x} \in S^{d-1}$ is a witness for C if $\mathbf{x}^T \mathbf{c} < 1$ for all $\mathbf{c} \in C$. We now show that with high probability, there is no witness for C in S^{d-1} . Using Lemma 3.8, it is sufficient to show that for any $\mathbf{x} \in N(\varepsilon)$, $\mathbf{x}^T \mathbf{c} \geq 2$ for some $\mathbf{c} \in C$, $\varepsilon \leq 1/R$.

Let us define E_1 to be the event that $\|\mathbf{c}\|^2 \leq R^2$ for all $\mathbf{c} \in C$. Since every entry of \mathbf{c} is chosen from i.i.d. Gaussian, the norm $\|\mathbf{c}\|^2$ is distributed according to χ^2 -distribution with variance $2d\sigma^4$. Since χ^2 -distribution is sub-exponential [123][Eq. 2.18], for any $\mathbf{c} \in C$, we have, for any $l \geq 1$, $\Pr(\|\mathbf{c}\|^2 > d\sigma^2(l+1)) \leq e^{-dl/8}$. This implies,

$$\Pr(\|\mathbf{c}\|^2 > R^2) \leq e^{-\frac{1}{8}(R^2/\sigma^2 - d)} \leq e^{-d},$$

substituting the value of σ^2 .

Then by union bound over all $\mathbf{c} \in C$.

$$\Pr[\bar{E}_1] \leq te^{-d} = e^{-d+20d/R^2+2 \log d} \leq e^{-\Omega(d)}, \quad (3.2)$$

for $R \geq 5$.

Let E_2 denote the event that for each $\mathbf{y} \in N(\varepsilon)$, there exists $\mathbf{c} \in C$ such that $\mathbf{y}^T \mathbf{c} \geq 2$. For any fixed $\mathbf{y} \in N(\varepsilon)$, and $\mathbf{c} \in C$, define $p_{\mathbf{y}, \mathbf{c}}$ to be the probability that $\mathbf{y}^T \mathbf{c} \geq 2$. Note that since c has i.i.d. Gaussian entries, then for any fixed $\mathbf{y} \in N(\varepsilon)$,

the inner product $\mathbf{y}^T \mathbf{c}$ is distributed according to $\mathcal{N}(0, \sigma^2)$. Using standard bounds for Gaussian distributions [16],

$$\begin{aligned} p_{\mathbf{y}, \mathbf{c}} &:= \Pr[\mathbf{y}^T \mathbf{c} \geq 2] \geq \frac{2\sigma}{(\sigma^2 + 4)\sqrt{2\pi}} e^{-\frac{2}{\sigma^2}} \\ &\geq \frac{1}{\sqrt{2\pi}} \min(\sigma^{-1}, \sigma/4) e^{-\frac{2}{\sigma^2}} \geq \frac{R}{12\sqrt{2\pi d}} e^{-\frac{2}{\sigma^2}}, \end{aligned}$$

for any $R \leq 6\sqrt{d}$.

Since each \mathbf{c} is chosen independently, the probability that $\mathbf{y}^T \mathbf{c} < 2$ for all $\mathbf{c} \in C$ is $(1 - p_{\mathbf{y}, \mathbf{c}})^t \leq e^{-t p_{\mathbf{y}, \mathbf{c}}}$. Now for $\varepsilon = 1/R$, by union bound, since $|N(\varepsilon)| \leq \left(\frac{3}{\varepsilon}\right)^d$,

$$\begin{aligned} \Pr[\bar{E}_2] &= \Pr[\exists \mathbf{y} \in N(\varepsilon) \text{ s.t. } \mathbf{y}^T \mathbf{c} < 2 \forall \mathbf{c} \in C] \\ &\leq e^{-t p_{\mathbf{y}, \mathbf{c}} + d \log 3R} \\ &= e^{-t \cdot e^{-\frac{18d}{R^2} - \log(\frac{12\sqrt{2\pi d}}{R})} + d \log 3R} \\ &= e^{-e^{\frac{2d}{R^2} + 2 \log d - \log(\frac{12\sqrt{2\pi d}}{R})} + d \log 3R} \\ &= e^{-d^2 e^{\frac{2d}{R^2} - \log(\frac{12\sqrt{2\pi d}}{R})} + d \log 3R} \\ &\leq e^{-\Omega(d)}. \end{aligned}$$

Therefore, $\Pr[\bar{E}_1 \cup \bar{E}_2] \leq e^{-\Omega(d)}$. Then using Lemma 3.8 and Theorem 3.6, we obtain the statement of the theorem. □

The above stated theorem provides a randomized algorithm to generate a point set of size $\exp(\Theta(d/R^2))$ such that the quantization scheme defined in Section 3.3 instantiated with this point set achieves a variance of $O(R^2)$ while communicating $O(d/R^2)$ bits, hence meeting the lower bound of Theorem 3.5. In particular, there exists a quantization scheme that achieves $O(1)$ variance with $O(d)$ bits of communication. Also, at the cost of communicating only $O(\log d)$ bits, our quantization scheme can

achieve a variance of $O(d/\log d)$. The deterministic constructions we provide, meet this bound up to a factor of $\log d$.

3.4.2 Derandomizing with Reed Muller Codes

In this section, we propose a deterministic construction of point set based on first order Reed-Muller codes that satisfy Condition 3.1. We assume d to be a power of 2, *i.e.*, $d = 2^p$ for some $p \geq 1$.

Our quantization scheme is based on the first order Reed-Muller codes, $\text{RM}(1, p)$ [87]. Each codeword of $\text{RM}(1, p)$ is given as the evaluation of a degree 1, p -variate polynomial over all points in \mathbb{F}_2^p . Mapping these codewords to reals using the coordinate-wise map $\phi : \mathbb{F}_2 \rightarrow \mathbb{R}$ defined as $\phi(b) = (-1)^b$ will give us a set of $2d$ points in $\{\pm 1\}^d$. Let RM denote this set of mapped codewords.

We show that the set of points in RM satisfy the characterization of Theorem 3.6, and therefore will give us a quantization scheme with $\log 2d$ communication and the following guarantees:

Proposition 3.9. *For any $\mathbf{v} \in B_d(\mathbf{0}_d, 1)$, let $\hat{\mathbf{v}} := Q_{C_{RM}}(\mathbf{v})$. Then, $\mathbb{E}\hat{\mathbf{v}} = \mathbf{v}$ and $\mathbb{E}[\|\mathbf{v} - \hat{\mathbf{v}}\|_2^2] = O(d)$.*

Proof of Proposition 3.9. We prove this theorem by showing that the point set C_{RM} satisfies the characterization of Theorem 3.6. Since all points in C_{RM} have squared norm exactly d , from Lemma 3.1 and Lemma 3.2, the proof follows.

First note that the matrix with the points in C_{RM} as its rows, has the following structure:

$$\mathbf{H} := \begin{bmatrix} \mathbf{H}_p \\ -\mathbf{H}_p \end{bmatrix}$$

where, \mathbf{H}_p is the $2^p \times 2^p$ Hadamard matrix.

For any fixed $\mathbf{x} \in S^{d-1}$, consider the sum $S(\mathbf{x}) := \sum_{\mathbf{c} \in C_{RM}} (\mathbf{x}^T \mathbf{c})^2$. We first show that $S(\mathbf{x}) \geq 2(d+1)$.

$$\begin{aligned} S(\mathbf{x}) &= \sum_{\mathbf{c} \in C_{RM}} (\mathbf{x}^T \mathbf{c})^2 = 2 \sum_{\mathbf{h}_i \in \mathbf{H}_p} (\mathbf{x}^T \mathbf{h}_i)^2 \\ &= 2 \|\mathbf{H}_p \mathbf{x}\|^2 = 2(\mathbf{x}^T \mathbf{H}_p^T)(\mathbf{H}_p \mathbf{x}) \\ &\stackrel{(i)}{=} 2d \cdot \|\mathbf{x}\|^2 = 2d. \end{aligned}$$

(i) follows from the fact that the columns of the Hadamard matrix are mutually orthogonal and therefore, $\mathbf{H}_p^T \mathbf{H}_p = d \cdot I_d$, where, I_d denotes the $d \times d$ identity matrix.

By an averaging argument, it then follows that there exists at least one $\mathbf{c} \in C_{RM}$ such that $|\mathbf{x}^T \mathbf{c}| \geq 1$. Since for every $\mathbf{c} \in C_{RM}$, there exists $-\mathbf{c} \in C_{RM}$, we get that $x^T \mathbf{c} \geq 1$ for some $\mathbf{c} \in C_{RM}$. \square

Remark 3.4. *Instead of first order Reed-Muller codes, we can use any binary linear code $C \subseteq \mathbb{F}_2^d$ to construct the point set as follows. Map all the codewords from \mathbb{F}_2^d to \mathbb{R}^d using ϕ described above. The point set containing all such mapped codewords, and their complements will give a quantization scheme with variance $O(d)$. The communication will however be $\log(2|C|)$, where $|C|$ denotes the number of codewords in C . In this regard, the first order Reed-Muller codes described above provide the best communication guarantees and is also efficiently computable.*

3.4.3 Other Deterministic Constructions

We now present several explicit constructions of point sets that give quantization schemes with varying tradeoffs. On one end of the spectrum, the cross-polytope scheme requires only $O(\log d)$ bits to communicate an unbiased estimate of a vector in \mathbb{R}^d with variance $O(d)$. While on the other end, the ε -net based scheme achieves a constant variance at the cost of $O(d)$ bits of communication.

3.4.3.1 Cross Polytope Scheme

Consider the following point set of $2d$ points in \mathbb{R}^d :

$$C_{cp} := \{\pm\sqrt{d} \mathbf{e}_i \mid i \in [d]\},$$

The convex hull $\text{CONV}(C_{cp})$ is a scaled cross polytope that satisfies Condition (3.1) with $R = \sqrt{d}$. Let $Q_{C_{cp}}$ be the instantiation of the quantization scheme described in Section 3.3 with the point set C_{cp} .

To compute the convex combination of any point $\mathbf{v} \in \text{CONV}(C_{cp})$, we need a non-negative solution to the following system of equations

$$\begin{bmatrix} \sqrt{d}I_d & -\sqrt{d}I_d \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_{2d} \end{bmatrix} = \begin{bmatrix} v_1 \\ \vdots \\ v_d \end{bmatrix} \quad \text{s.t.} \quad \sum_{i=1}^{2d} a_i = 1, \quad (3.3)$$

where, I_d is the $d \times d$ identity matrix. Equation 3.3 leads to the following closed form solution that can be computed in $O(d)$ time:

$$a_i = \begin{cases} \frac{v_i}{\sqrt{d}} + \frac{\gamma}{2d} & \text{if } v_i > 0 \text{ and } i \leq d \\ -\frac{v_i}{\sqrt{d}} + \frac{\gamma}{2d} & \text{if } v_i \leq 0 \text{ and } i > d \\ \frac{\gamma}{2d} & \text{otherwise} \end{cases} \quad (3.4)$$

where, $\gamma := 1 - \frac{\|\mathbf{v}\|_1}{\sqrt{d}}$, is a non-negative quantity for every $\mathbf{v} \in B_d(\mathbf{0}_d, 1)$.

The bound on the variance of the quantizer follows directly from Lemma 3.2.

Proposition 3.10. *For any $\mathbf{v} \in B_d(\mathbf{0}_d, 1)$, let $\hat{\mathbf{v}} := Q_{C_{cp}}(\mathbf{v})$. Then, $\mathbb{E}[\hat{\mathbf{v}}] = \mathbf{v}$ and $\mathbb{E}[\|\mathbf{v} - \hat{\mathbf{v}}\|_2^2] = O(d)$.*

Proof of Proposition 3.10. The proof of Proposition 3.10 follows directly from Lemma 3.2 provided the point set C_{cp} satisfies Condition (3.1) with $R = \sqrt{d}$. We will now prove this fact.

Since each vertex is of the form $\pm\sqrt{d}\mathbf{e}_i$, it follows that all the vertices of $\text{CONV}(C_{cp})$, and hence the entire convex hull lies inside a ball of radius \sqrt{d} , i.e., $\text{CONV}(C_{cp}) \subset B_d(\mathbf{0}_d, \sqrt{d})$.

To prove that the unit ball is contained in the convex hull $\text{CONV}(C_{cp})$, we pick any arbitrary point $v \in B_d(\mathbf{0}_d, 1)$ and show that it can be written as a convex combination of points in C_{cp} . The fact follows from the solution to the system of linear equations (3.3) given in Equation (3.14). Note that the solution satisfies $a_i \geq 0$ and $\sum_i a_i = 1$ for any point $\mathbf{v} \in B_d(0, 1)$. \square

Moreover, using the variance reduction technique described in Section 3.3 with $s = O(\frac{d}{\log d})$, the cross polytope based quantization $Q_{C_{cp}}$ achieves a variance of $O(\log d)$ at the cost of communicating $O(d)$ bits.

We note that the cross-polytope quantization scheme described above when used along with the variance reduction technique (by repetition), is in essence similar to Maurey sparsification ([3]).

3.4.3.2 Scaled ε -nets

On the other end of the spectrum, we now show the existence of point sets of exponential size that are contained in a constant radius ball. This point set allows us to obtain a gradient quantization scheme with $O(d)$ communication and $O(1)$ variance. We show that an appropriate constant scaling of an ε -net points satisfies Condition (3.1).

Lemma 3.11. *For any $0 < \varepsilon < 1$, let $R = \frac{1}{1-\varepsilon}$. The point set $C_{net} := \{R \cdot \mathbf{u} \mid \mathbf{u} \in N(\varepsilon)\}$ satisfies Condition (3.1).*

Proof of Lemma 3.11. Let $K := \text{CONV}(N(\varepsilon))$ be the convex hull of the ε -net points of the unit sphere. Let $B_d(\mathbf{0}_d, r)$ be the inscribed ball in K for some $r < 1$. We show that $r \geq 1 - \varepsilon$.

Consider the face of K that is tangent to $B_d(\mathbf{0}_d, r)$ at point \mathbf{z} . We will show that $\|\mathbf{z}\|_2 \geq 1 - \varepsilon$. Extend the line joining $(\mathbf{0}_d, \mathbf{z})$ to meet S^{d-1} at point \mathbf{x} . Since $\mathbf{x} \in \mathcal{S}^{d-1}$, we know that there exists a net point \mathbf{u} at a distance of at most ε from it. Therefore, the distance of \mathbf{x} from K is upper bounded by ε , *i.e.*, $\text{dist}(\mathbf{x}, K) = \|\mathbf{x} - \mathbf{z}\| \leq \|\mathbf{x} - \mathbf{u}\| \leq \varepsilon$. Therefore $\|\mathbf{z}\| = 1 - \|\mathbf{x} - \mathbf{z}\| \geq 1 - \varepsilon$.

Therefore scaling all the points of $N(\varepsilon)$ by any $R \geq \frac{1}{1-\varepsilon}$ we see that $B_d(\mathbf{0}_d, 1) \subseteq \text{CONV}(C)$. \square

Let Q_{net} be the instantiation of vqSGD with point set C_{net} . From Lemma 3.2, we then get the following guarantees for the quantization scheme obtained from scaled ε -nets, C_{net} for some constant $\varepsilon < 1$.

Proposition 3.12. *For any $\mathbf{v} \in B_d(\mathbf{0}_d, 1)$, let $\hat{\mathbf{v}} := Q_{C_{\text{net}}}(\mathbf{v})$. Then, $\mathbb{E}[\hat{\mathbf{v}}] = \mathbf{v}$ and $\mathbb{E}[\|\mathbf{v} - \hat{\mathbf{v}}\|_2^2] = \frac{1}{(1-\varepsilon)}$.*

Moreover, Q_{net} requires $O(d \log \frac{1}{\varepsilon})$ bits to represent the unbiased gradient estimate.

3.5 Private Quantization

In this section we show that under certain conditions the quantization scheme $Q_C(\cdot)$ obtained from the point set C is also ϵ -differentially private. First, we see why the quantization scheme described in Section 3.3 is not privacy preserving in general.

Let C be any point set with $|C| > d + 1$. For any point $\mathbf{x} = \sum_{i=1}^{|C|} a_i \mathbf{c}_i \in \text{CONV}(C)$, let $\text{supp}(\mathbf{x})C = \{\mathbf{c}_i \in C \mid a_i \neq 0\}$ denote the points in C that are in the range of $Q_C(\mathbf{x})$.

In order for Q_C to be ϵ -DP for any $\epsilon > \epsilon_0$, we have to show for gradients $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ of any two neighboring datasets and for any $\mathbf{z} \in \text{supp}(\mathbf{x})C \cup \text{supp}(\mathbf{y})C$,

$$\Pr[Q_C(\mathbf{x}) = \mathbf{z}] \leq e^{\epsilon_0} \cdot \Pr[Q_C(\mathbf{y}) = \mathbf{z}]. \quad (3.5)$$

If $|C| > d + 1$, there may exist two gradients $\mathbf{x}, \mathbf{y} \in \text{CONV}(C)$ such that $\text{supp}(\mathbf{x})C \neq \text{supp}(\mathbf{y})C$. Therefore, for $\mathbf{z} \in \text{supp}(\mathbf{x})C \setminus \text{supp}(\mathbf{y})C$, Eq. (3.5) will not hold for any finite ϵ_0 .

The discussion above establishes a sufficient condition for the quantization scheme Q_C to be differentially private. Essentially, we want all points in $B_d(\mathbf{0}_d, 1)$ to have full support on all the points in C . This is definitely possible when $|C| = d + 1$. Therefore if the point set satisfying Condition (3.1) has size $|C| = d + 1$, then the quantization scheme Q_C is ϵ -differentially private, for some $\epsilon > \epsilon(C)$.

We now present two constructions of point sets C of size exactly $d + 1$ satisfying Condition (3.1) that give an ϵ -differentially private quantization scheme. Both the schemes achieve a communication cost of $\log(d + 1)$, but the variance is a factor d larger than the non-private scheme, $Q_{C_{cp}}$.

(1) Simplex Scheme: Consider the following set of $d + 1$ points

$$C_S = \{2d \mathbf{e}_i \mid i \in [d]\} \cup \{-4\mathbf{1}_d\}.$$

The convex hull of C_S satisfies Condition (3.1) with $R = O(d)$. Since the size of the set is exactly $d + 1$, every point in the unit ball can be represented as a convex combination of all the points in C_S (*i.e.*, all coefficients of the convex combination are non zero). This fact will be used crucially to show that this scheme is also differentially private.

The coefficients of the convex combination of any point $\mathbf{v} \in \text{CONV}(C_S)$ can be computed from the following system of linear equations:

$$\begin{bmatrix} -4\mathbf{1}_d^T & 2\sqrt{d}I_d \end{bmatrix} \begin{bmatrix} a_0 & \dots & a_d \end{bmatrix}^T = \begin{bmatrix} v_1 & \dots & v_d \end{bmatrix}^T \text{ such that } \sum_{i=0}^d a_i = 1. \quad (3.6)$$

Equation 3.6 leads to the following closed form solution that can be computed in linear-time:

$$a_0 = 1/3 - (\sum_{i=1}^d v_i)/(6d) \quad a_i = v_i/(2d) + 2a_0/d \quad \forall i \geq 1. \quad (3.7)$$

Proposition 3.13. *For any $\mathbf{v} \in B_d(\mathbf{0}_d, 1)$, let $\hat{\mathbf{v}} := Q_{C_S}(\mathbf{v})$. Then, $\mathbb{E}[\hat{\mathbf{v}}] = \mathbf{v}$ and $\mathbb{E}[\|\mathbf{v} - \hat{\mathbf{v}}\|_2^2] = O(d^2)$. Moreover, Q_{C_S} is ϵ -DP for any $\epsilon > \log 7$.*

Proof of Proposition 3.13. First we show that the point set C_S satisfies Condition (3.1) with $R = 2d$. The fact that $\text{CONV}(C_S) \subset B_d(\mathbf{0}_d, 2d)$ follows trivially from the observation that each point in $C_S \in B_d(\mathbf{0}_d, 2d)$.

To show that $B_d(\mathbf{0}_d, 1) \subset \text{CONV}(C_S)$, consider any face of the convex hull, $F_{\mathbf{c}} := \text{CONV}(C_S \setminus \{\mathbf{c}\})$, for some $\mathbf{c} \in C_S$. We show that $F_{\mathbf{c}}$ is at an ℓ_2 distance of at least 1 from $\mathbf{0}_d$. This in turn shows that any point outside the convex hull must be outside the unit ball as well.

First consider the case when $\mathbf{c} = -4\mathbf{1}_d$. We observe that the face $F_{\mathbf{c}}$ is contained in the hyper-plane $H_{\mathbf{c}} := \{\mathbf{x} \in \mathbb{R}^d \mid \frac{1}{\sqrt{d}}\mathbf{1}_d^T \mathbf{x} = 2\sqrt{d}\}$, and therefore is at a distance of $O(\sqrt{d})$ from the origin.

Now consider the case when $\mathbf{c} = 2d\mathbf{e}_1$. Let $\mathbf{w} = \frac{1}{\sqrt{\frac{9}{16} - \frac{1}{2d}}}(-\frac{3}{4} + \frac{1}{2d}, \frac{1}{2d}, \dots, \frac{1}{2d})^T \in \mathbb{R}^d$ be a unit vector. We note that $F_{\mathbf{c}} \subset H_{\mathbf{c}}$, where $H_{\mathbf{c}} := \{\mathbf{x} \in \mathbb{R}^d \mid \mathbf{w}^T \mathbf{x} = \frac{1}{\sqrt{\frac{9}{16} - \frac{1}{2d}}}\}$ is the hyper-plane defined by the unit normal vector \mathbf{w} that is at a distance of at least 1 from $\mathbf{0}_d$.

Since all other faces are symmetric, the proof for the case $\mathbf{c} = 2d\mathbf{e}_i, i \in [d]$ follows similarly.

Privacy: We now show that the quantization scheme is ϵ -differentially private for any $\epsilon > \log 7$. From the definition of ϵ -DP, it is sufficient to show that for any $\mathbf{x}, \mathbf{y} \in B_d(\mathbf{0}_d, 1)$, and every $\mathbf{c} \in C_S$,

$$\frac{\Pr[Q_{C_S}(\mathbf{x}) = \mathbf{c}]}{\Pr[Q_{C_S}(\mathbf{y}) = \mathbf{c}]} \leq 7.$$

Since $\mathbf{x}, \mathbf{y} \in \text{CONV}(C_S)$, we can express them as the convex combination of points in C_S . Let $\mathbf{x} = \sum_{\mathbf{c} \in C_S} a_c^{(\mathbf{x})} \mathbf{c}$. Similarly, let $\mathbf{y} = \sum_{\mathbf{c} \in C_S} a_c^{(\mathbf{y})} \mathbf{c}$. Then, from the construction of the quantization function Q_{C_S} , we know that

$$\frac{\Pr[Q_{C_S}(\mathbf{x}) = \mathbf{c}]}{\Pr[Q_{C_S}(\mathbf{y}) = \mathbf{c}]} = \frac{a_c^{(\mathbf{x})}}{a_c^{(\mathbf{y})}}.$$

We now show that the ratio $\frac{a_c^{(\mathbf{x})}}{a_c^{(\mathbf{y})}}$ is at most 7 for *any* pair $\mathbf{x}, \mathbf{y} \in B_d(\mathbf{0}_d, 1)$ and any $\mathbf{c} \in C_S$. The privacy bound follows from this observation.

First, consider the case $\mathbf{c} = -4\mathbf{1}_d$. From the closed form solution for any $\mathbf{x} \in \text{CONV}(C_S)$ described in Equation (3.7), we know that $a_c^{(\mathbf{x})} = \frac{1}{3} - \frac{\sum_{i=1}^d x_i}{6d}$. For any $\mathbf{x} \in B_d(\mathbf{0}_d, 1)$, $\sum_{i=1}^d x_i \in [-\|\mathbf{x}\|_1, \|\mathbf{x}\|_1] \subseteq [-\sqrt{d}, \sqrt{d}]$. Therefore, $a_c^{(\mathbf{x})} \in \left[\frac{1}{3} - \frac{1}{6\sqrt{d}}, \frac{1}{3} + \frac{1}{6\sqrt{d}}\right]$. It then follows that for any $\mathbf{x}, \mathbf{y} \in B_d(\mathbf{0}_d, 1)$ and $\mathbf{c} = -4\mathbf{1}_d$,

$$\frac{a_c^{(\mathbf{x})}}{a_c^{(\mathbf{y})}} \leq \frac{\frac{1}{3} + \frac{1}{6\sqrt{d}}}{\frac{1}{3} - \frac{1}{6\sqrt{d}}} = 1 + \frac{2}{2\sqrt{d} - 1} \leq 3$$

Now we consider the case when $\mathbf{c} = 2d \mathbf{e}_1$. Then from the closed form solution in Equation (3.7), we get that for any $\mathbf{x} \in \text{CONV}(C_S)$ the coefficient $a_c^{(\mathbf{x})} = \frac{x_1}{2d} \left(1 - \frac{2}{3d}\right) - \frac{\sum_{i=2}^d x_i}{3d^2} + \frac{2}{3d}$. Note that this quantity is maximized for $\mathbf{x} = \mathbf{e}_1$ and minimized for $\mathbf{x} = -\mathbf{e}_1$. Therefore the ratio for any $\mathbf{x}, \mathbf{y} \in B_d(\mathbf{0}_d, 1)$ and $\mathbf{c} = 2d \mathbf{e}_1$ is at most

$$\frac{a_c^{(\mathbf{x})}}{a_c^{(\mathbf{y})}} \leq \frac{7d - 2}{d + 2} \leq 7$$

The ratio for all other vertices can be computed in a similar fashion and is bounded by the same quantity. \square

(2) Hadamard Scheme: We now propose another quantization scheme with same communication cost, but provides better privacy guarantees. This quantization

scheme is similar to the one presented in Section 3.4.2 and is based on the columns of a Hadamard matrix.

Let us assume that $d + 1$ is a power of 2, *i.e.*, $d + 1 = 2^p$ for some $p \geq 1$. For any $i \in [d + 1]$, let $\mathbf{h}_i \in \mathbb{R}^d$ denote the i -th column of \mathbf{H}_p with the first coordinate punctured. Consider the following set of $d + 1$ points obtained from the punctured columns of H_p :

$$C_H = \{2\sqrt{d} \mathbf{h}_i \mid i \in [d + 1]\}$$

The quantization scheme Q_{C_H} can be implemented in linear time since computing the probabilities requires computing a matrix vector product,

$$(d + 1) \cdot \begin{bmatrix} a_1 & \cdots & a_{d+1} \end{bmatrix}^T = H_p^T \begin{bmatrix} 1 & \mathbf{v}^T / (2\sqrt{d}) \end{bmatrix}^T$$

that has closed form solution for each a_i as:

$$a_i = (1 + (\mathbf{h}_i^T \mathbf{v}) / (2\sqrt{d})) / (d + 1) \quad (3.8)$$

Proposition 3.14. *For any $\mathbf{v} \in B_d(\mathbf{0}_d, 1)$, let $\hat{\mathbf{v}} := Q_{C_H}(\mathbf{v})$. Then, $\mathbb{E}[\hat{\mathbf{v}}] = \mathbf{v}$ and $\mathbb{E}[\|\mathbf{v} - \hat{\mathbf{v}}\|_2^2] = O(d^2)$. Moreover, Q_{C_H} is ϵ -DP for any $\epsilon > \log(1 + \sqrt{2})$.*

Proof of Proposition 3.14. First, we show that C_H satisfies Condition (3.1) with $R = 2d$. The fact that $\text{CONV}(C_H) \subset B_d(\mathbf{0}_d, 2d)$ is trivial and follows since every point in C_H is contained in $B_d(\mathbf{0}_d, 2d)$.

To show that $B_d(\mathbf{0}_d, 1) \subset C_H$, consider any $\mathbf{x} \in B_d(\mathbf{0}_d, 1)$, and the closed form solution for the coefficients a_i given by Equation (3.8). We now show that these coefficients indeed give a convex combination. Note that $a_i := \frac{1}{d+1} \left(1 + \frac{\mathbf{c}^T \mathbf{x}}{4d}\right) \geq 0$.

This holds since $\mathbf{c}^T \mathbf{x} \geq \|\mathbf{c}\| \|\mathbf{x}\| \geq -2d$. Moreover, from the property of Hadamard matrices,

$$\sum_{i=1}^{d+1} a_i = \frac{1}{d+1} \begin{bmatrix} 1 & \dots & 1 \end{bmatrix} \mathbf{H}_p^T \begin{bmatrix} 1 \\ \frac{\mathbf{x}}{2\sqrt{d}} \end{bmatrix} = 1.$$

The last equality follows from the following property of the Hadamard matrices that can be proved using induction.

$$\begin{bmatrix} 1 & \dots & 1 \end{bmatrix} \mathbf{H}_p^T = \begin{bmatrix} 2^p & 0 & \dots & 0 \end{bmatrix}.$$

Therefore, any $\mathbf{x} \in B_d(\mathbf{0}_d, 1)$ can be expressed as a convex combination of the points in C_H , i.e., $\mathbf{x} = \sum_{i=1}^{d+1} a_i \mathbf{c}_i$, for $\mathbf{c}_i \in C_H$.

Privacy: We now show that the quantization scheme is ϵ -differentially private for any $\epsilon > 0.4$. From the definition of ϵ -DP, it is sufficient to show that for any $\mathbf{x}, \mathbf{y} \in B_d(\mathbf{0}_d, 1)$, and any $\mathbf{c} \in C_H$,

$$\frac{\Pr[Q_{C_H}(\mathbf{x}) = \mathbf{c}]}{\Pr[Q_{C_S}(\mathbf{y}) = \mathbf{c}]} \leq 1 + \sqrt{2}$$

Since $\mathbf{x}, \mathbf{y} \in \text{CONV}(C_S)$, we can express them as the convex combination of points in C_H . Let $\mathbf{x} = \sum_{\mathbf{c} \in C_H} a_c^{(\mathbf{x})}$. Similarly, let $\mathbf{y} = \sum_{\mathbf{c} \in C_H} a_c^{(\mathbf{y})}$. Then, from the construction of the quantization function Q_{C_H} , we know that

$$\frac{\Pr[Q_{C_H}(\mathbf{x}) = \mathbf{c}]}{\Pr[Q_{C_H}(\mathbf{y}) = \mathbf{c}]} = \frac{a_c^{(\mathbf{x})}}{a_c^{(\mathbf{y})}}. \quad (3.9)$$

From the closed form solution in Equation (3.8), we know that for any $\mathbf{x} \in \text{CONV}(C_H)$, the coefficient of \mathbf{c} in the convex combination of \mathbf{x} is given by $a_c^{(\mathbf{x})} = \frac{1}{d+1} \left(1 + \frac{\mathbf{c}^T \mathbf{x}}{4d} \right)$. Plugging this in Equation (3.9), we get

$$\frac{\Pr[Q_{C_H}(\mathbf{x}) = \mathbf{c}]}{\Pr[Q_{C_H}(\mathbf{y}) = \mathbf{c}]} = \frac{a_c^{(\mathbf{x})}}{a_c^{(\mathbf{y})}} = \frac{1 + \frac{\mathbf{c}^T \mathbf{x}}{4d}}{1 + \frac{\mathbf{c}^T \mathbf{y}}{4d}} = 1 + \frac{\frac{\mathbf{c}^T (\mathbf{x} - \mathbf{y})}{4d}}{1 + \frac{\mathbf{c}^T \mathbf{y}}{4d}} \quad (3.10)$$

$$\leq 1 + \frac{\|\mathbf{c}\|_2 \|\mathbf{x} - \mathbf{y}\|_2}{4d - \|\mathbf{c}\|_2} \quad \text{for } \mathbf{y} = -\frac{\mathbf{c}}{\|\mathbf{c}\|_2} \quad (3.11)$$

$$\leq 1 + \frac{2\sqrt{2}d}{4d - 2d} \quad (3.12)$$

(since $\|\mathbf{x} - \mathbf{y}\|_2 \leq \sqrt{2}$ and $\|\mathbf{c}\|_2 = 2d$.)

$$= 1 + \sqrt{2} \quad (3.13)$$

This concludes the proof of Proposition 3.14. \square

Finally, we remark that even though the point set C_{cp} in the cross-polytope scheme has more than $d + 1$ points, it still gives us ϵ -DP - although with slightly worse privacy guarantee.

Proposition 3.15. $Q_{C_{cp}}$ is ϵ -DP for any $\epsilon > \log d$.

Proof of Proposition 3.15. The fact that $Q_{\tilde{C}_{cp}}$ satisfies Condition (3.1) with $R = 2\sqrt{d}$ follows from the proof of Proposition 3.10. For any $v \in \mathbb{R}^d$, we can compute the convex combinations as

$$a_i = \begin{cases} \frac{v_i}{2\sqrt{d}} + \frac{\gamma}{2d} & \text{if } v_i > 0 \text{ and } i \leq d \\ -\frac{v_i}{2\sqrt{d}} + \frac{\gamma}{2d} & \text{if } v_i \leq 0 \text{ and } i > d \\ \frac{\gamma}{2d} & \text{otherwise} \end{cases} \quad (3.14)$$

where, $\gamma := 1 - \frac{\|\mathbf{v}\|_1}{2\sqrt{d}}$, is a non-negative quantity for every $\mathbf{v} \in B_d(\mathbf{0}_d, 1)$.

To prove the privacy guarantees of this scheme, we first state a few observations:

- Since $\|\mathbf{v}\|_1 \in [-\sqrt{d}, \sqrt{d}]$, the quantity $\gamma \in [1/2, 3/2]$.
- For any coordinate $i \in [d]$, if $x_i > 0$, then the coefficients $a_i = \frac{|x_i|}{2\sqrt{d}} + \frac{\gamma}{2d}$, and $a_{d+i} = \frac{\gamma}{2d}$.

- Similarly, if $x_i \leq 0$, then the coefficients $a_i = \frac{\gamma}{2d}$, and $a_{d+i} = \frac{|x_i|}{2\sqrt{d}} + \frac{\gamma}{2d}$.
- For any $\mathbf{x} \in B_d(\mathbf{0}_d, 1)$, $x_i \in [-1, 1]$

Let $\mathbf{x}, \mathbf{y} \in B_d(\mathbf{0}_d, 1)$ and for any $\mathbf{c} \in \tilde{C}_{cp}$, we need to upper bound the following quantity to prove the privacy guarantees of the scheme:

$$p_c := \frac{\Pr[Q_{\tilde{C}_{cp}}(\mathbf{x}) = \mathbf{c}]}{\Pr[Q_{\tilde{C}_{cp}}(\mathbf{y}) = \mathbf{c}]}$$

Note that it is sufficient to consider only one of the points $\mathbf{c} = 2\sqrt{d}\mathbf{e}_j$ in the following four scenarios:

1. $x_i > 0, y_i > 0$, then $p_c = \frac{\frac{|x_i|}{2\sqrt{d}} + \frac{\gamma x}{2d}}{\frac{|y_i|}{2\sqrt{d}} + \frac{\gamma y}{2d}} \leq \frac{\frac{1}{2\sqrt{d}} + \frac{3}{4d}}{\frac{1}{4d}} \leq O(\sqrt{d})$.
2. $x_i > 0, y_i \leq 0$, then $p_c = \frac{\frac{|x_i|}{2\sqrt{d}} + \frac{\gamma x}{2d}}{\frac{\gamma y}{2d}} \leq \frac{\frac{1}{2\sqrt{d}} + \frac{3}{4d}}{\frac{1}{4d}} \leq O(\sqrt{d})$.
3. $x_i \leq 0, y_i > 0$, then $p_c = \frac{\frac{\gamma x}{2d}}{\frac{|y_i|}{2\sqrt{d}} + \frac{\gamma y}{2d}} = \frac{\frac{3}{4d}}{\frac{1}{4d}} \leq 3$
4. $x_i \leq 0, y_i \leq 0$, then $p_c = \frac{\frac{\gamma x}{2d}}{\frac{\gamma y}{2d}} = \frac{\frac{3}{4d}}{\frac{1}{4d}} \leq 3$.

Therefore, the privacy guarantees hold for any $\epsilon > O(\log d)$. \square

We now show a Randomized Response (RR) scheme that can be used on top of any of our quantization schemes to achieve privacy. This scheme incurs the same communication as the original quantizer, however, the price of privacy is paid by factor of d increase in the variance. We also propose a weaker version using RAPPOR, that incurs a higher communication cost depending on the point set of choice.

3.5.1 Randomized Response

We present a Randomized Response (RR) [130] mechanism that can be used over the output of Q_C to make it ϵ -DP (for any $\epsilon > 0$). This modified scheme retains the original communication cost of Q_C , but the cost for privacy is paid by a factor of $O(d)$ in the variance term.

Recall that the quantization scheme described in Section 3.3, $Q_C(\mathbf{v})$, takes a vector $\mathbf{v} \in B_d(\mathbf{0}_d, 1)$ and returns a point $\mathbf{c}_i \in C$. The RR scheme takes the output of $Q_C(\mathbf{v})$ and returns another random vector from C .

For any $\epsilon > 0$, define $p := p(\epsilon) = \frac{e^\epsilon}{e^\epsilon + |C| - 1}$ and $q := \frac{1-p}{|C|-1} = \frac{1}{e^\epsilon + |C| - 1}$. We define the private quantization of a vector $\mathbf{v} \in B_d(\mathbf{0}_d, 1)$ as

$$\hat{\mathbf{v}} = PQ_{C,\epsilon}(\mathbf{v}) = \frac{1}{p-q} \sum_{i=1}^{|C|} (\mathbf{1}_{\{\mathbf{y}=\mathbf{c}_i\}} - q) \mathbf{c}_i,$$

where, $\mathbf{1}_{\{\mathbf{y}=\mathbf{c}_i\}}$ is an indicator of the event $\mathbf{y} = \mathbf{c}_i$ and $\mathbf{y} := \text{RR}_p(Q_C(\mathbf{v}), C)$ is defined as

$$\text{RR}_p(Q_C(\mathbf{v}), C) = \begin{cases} Q_C(\mathbf{v}) & \text{w.p. } p \\ \mathbf{z} \in C \setminus \{Q_C(\mathbf{v})\} & \text{w.p. } q \end{cases}$$

We claim that the quantization scheme $PQ_{C,\epsilon}$ is ϵ -differentially private.

Theorem 3.16. *Let $C \subset \mathbb{R}^d$ be any point set satisfying Condition (3.1). For any $\epsilon > 0$, let $p = \frac{e^\epsilon}{e^\epsilon + |C| - 1}$ and $q = \frac{1}{e^\epsilon + |C| - 1}$. For any $\mathbf{v} \in B_d(\mathbf{0}_d, 1)$, let $\hat{\mathbf{v}} = PQ_{C,\epsilon}(\mathbf{v}) = \frac{1}{p-q} \sum_{i=1}^{|C|} (\mathbf{1}_{\{\mathbf{y}=\mathbf{c}_i\}} - q) \mathbf{c}_i$, where, $\mathbf{y} := \text{RR}_p(Q_C(\mathbf{v}), C)$. Then, $\mathbb{E}[\hat{\mathbf{v}}] = \mathbf{v}$ and $\mathbb{E}[\|\mathbf{v} - \hat{\mathbf{v}}\|_2^2] = O(|C|R^2)$, where the expectation is taken over the randomness in both Q_C and RR_p . Moreover, the scheme is ϵ -differentially private.*

Proof of Theorem 3.16. First we show that $\hat{\mathbf{v}} = PQ_{C,\epsilon}(\mathbf{v}) = \frac{1}{p-q} \sum_{i=1}^{|C|} (\mathbf{1}_{\{\mathbf{y}=\mathbf{c}_i\}} - q) \mathbf{c}_i$ is an unbiased estimator of \mathbf{v} . From linearity of expectations, we have

$$\mathbb{E}[\hat{\mathbf{v}}] = \frac{1}{p-q} \sum_{i=1}^{|C|} (\Pr[\mathbf{y} = \mathbf{c}_i] - q) \mathbf{c}_i, \quad (3.15)$$

where, the expectation is taken over the randomness of both the quantization and RR scheme. Recall that

$$\mathbf{y} := \text{RR}_p(Q_C(\mathbf{v}), C) \in C,$$

where $p = \frac{e^\epsilon}{e^\epsilon + |C| - 1}$. Therefore,

$$\begin{aligned}\Pr(\mathbf{y} = \mathbf{c}_i) &= \sum_{j=1}^{|C|} \Pr[\mathbf{y} = \mathbf{c}_i | Q_C(\mathbf{v}) = \mathbf{c}_j] \cdot \Pr[Q_C(\mathbf{v}) = \mathbf{c}_j] \\ &= (p - q)a_i + q.\end{aligned}$$

Therefore $\mathbb{E}[\hat{\mathbf{v}}] = \frac{1}{p-q} \sum_{i=1}^{|C|} a_i \mathbf{c}_i = \mathbf{v}$.

Now we bound the variance of the estimator

$$\begin{aligned}\mathbb{E}[\|\mathbf{v} - \hat{\mathbf{v}}\|^2] &= \mathbb{E}[\|\sum_{i=1}^{|C|} (\frac{1}{p-q}(\mathbf{1}_{\{\mathbf{y}=\mathbf{c}_i\}} - q) - a_i) \mathbf{c}_i\|^2] \\ &\leq \sum_{i=1}^{|C|} \mathbb{E}[\|(\frac{1}{p-q}(\mathbf{1}_{\{\mathbf{y}=\mathbf{c}_i\}} - q) - a_i)^2 \|\mathbf{c}_i\|^2] \\ &= \sum_{i=1}^{|C|} \text{Var}[\|(\frac{1}{p-q}(\mathbf{1}_{\{\mathbf{y}=\mathbf{c}_i\}} - q))\| \|\mathbf{c}_i\|^2] \\ &= (\frac{1}{p-q})^2 \sum_{i=1}^{|C|} \text{Var}(\mathbf{1}_{\{\mathbf{y}=\mathbf{c}_i\}}) \|\mathbf{c}_i\|^2 \\ &= O(|C|R^2)\end{aligned}$$

As $\|\mathbf{c}_i\|^2 \leq R^2$ and $\text{Var}(\mathbf{1}_{\{\mathbf{y}=\mathbf{c}_i\}}) \leq 1/4$.

Privacy Now we show that our scheme is ϵ differentially private where ϵ is the input parameter to the RR algorithm. For any two points $\mathbf{v}, \mathbf{w} \in B_d(\mathbf{0}_d, 1)$,

$$\frac{PQ_{C,\epsilon}(\mathbf{v}) = y}{PQ_{C,\epsilon}(\mathbf{w}) = y} = \frac{\sum_{i=1}^{|C|} \Pr(y|Q_C(\mathbf{v}) = \mathbf{c}_i) \Pr(Q_C(\mathbf{v}) = \mathbf{c}_i)}{\sum_{j=1}^{|C|} \Pr(y|Q_C(\mathbf{w}) = \mathbf{c}_j) \Pr(Q_C(\mathbf{w}) = \mathbf{c}_j)} \quad (3.16)$$

$$\leq \frac{\max_i \Pr(y|Q_C(\mathbf{v}) = \mathbf{c}_i) \sum_{i=1}^{|C|} \Pr(Q_C(\mathbf{v}) = \mathbf{c}_i)}{\min_j \Pr(y|Q_C(\mathbf{w}) = \mathbf{c}_j) \sum_{i=1}^{|C|} \Pr(Q_C(\mathbf{w}) = \mathbf{c}_j)} \quad (3.17)$$

$$= \frac{\max_i \Pr(y|Q_C(\mathbf{v}) = \mathbf{c}_i)}{\min_j \Pr(y|Q_C(\mathbf{w}) = \mathbf{c}_j)} \leq e^\epsilon \quad (3.18)$$

we are using the following privacy property of Randomized Rounding [130] mechanism in Equation (3.18)

$$\sup_{i,j} \frac{\Pr(y|Q_C(\mathbf{v}) = \mathbf{c}_i)}{\Pr(y|Q_C(\mathbf{w}) = \mathbf{c}_j)} \leq e^\epsilon \quad \forall \mathbf{v}, \mathbf{w}$$

□

3.5.2 Privacy Using RAPPOR

In this section, we present an alternate mechanism to make the quantization scheme ϵ -DP (for any $\epsilon > 0$). The main idea is to use the RAPPOR [45] mechanism over a 1-hot encoding of the indices of vertices in C . Though in doing so, we have to tradeoff on the communication a bit. Instead of sending $\log |C|$ bits, this scheme now requires one to send $O(|C|)$ bits to achieve privacy.

Recall that the quantization scheme described in Section 3.3, $Q_C(\mathbf{v})$, takes a vector $\mathbf{v} \in B_d(\mathbf{0}_d, 1)$ and returns a point \mathbf{c}_i in C . We can interpret the output as the bit string $\mathbf{b} \in \{0, 1\}^{|C|}$ which is the indicator of the point \mathbf{c}_i in C (according to some fixed arbitrary ordering of C). Note that this is essentially the 1-hot encoding of \mathbf{c}_i . In the RAPPOR scheme each bit of the 1-hot bit string \mathbf{b} is flipped independently with probability $p := p(\epsilon) = \frac{1}{(e^{\epsilon/2} + 1)}$.

For any $\epsilon > 0$, let $p = \frac{1}{(e^{\epsilon/2} + 1)}$. Define, the private quantization of a vector $\mathbf{v} \in B_d(\mathbf{0}_d, 1)$ as

$$\hat{\mathbf{v}} := PQ_{C,\epsilon}(\mathbf{v}) = \frac{1}{(1 - 2p)} \sum_{j=1}^{|C|} (y_j - p) \mathbf{c}_j$$

where, $\mathbf{y} := \text{RAPPOR}_p(1\text{-HOT}(Q_C(\mathbf{v}), C)) \in \{0, 1\}^{|C|}$.

We claim that the quantization scheme $PQ_{C,\epsilon}$ is ϵ -differentially private. Moreover, adding the noise over the 1-HOT encoding maintains the unbiasedness of the gradient estimate but incurs a factor of $|C|$ in variance term while the communication cost is $O(|C|)$.

Theorem 3.17. *Let $C \subset \mathbb{R}^d$ be any point set satisfying Condition (3.1). For any $\epsilon > 0$, let $p = \frac{1}{(e^{\epsilon/2} + 1)}$. For any $\mathbf{v} \in B_d(\mathbf{0}_d, 1)$, let $\hat{\mathbf{v}} := \frac{1}{1-2p} \sum_{j=1}^{|C|} (y_j - p) \mathbf{c}_j$, where, $\mathbf{y} := \text{RAPPOR}_p(1\text{-HOT}(Q_C(\mathbf{v}), C))$. Then, $\mathbb{E}[\hat{\mathbf{v}}] = \mathbf{v}$ and $\mathbb{E}[\|\mathbf{v} - \hat{\mathbf{v}}\|_2^2] = O(|C|R^2)$. Moreover, the scheme is ϵ -differentially private.*

Proof of Theorem 3.17. First we show that $\hat{\mathbf{v}} = \frac{1}{(1-2p)} \sum_{j=1}^{|C|} (y_j - p) \mathbf{c}_j$ is an unbiased estimator of \mathbf{v} . From linearity of expectations, we have

$$\mathbb{E}[\hat{\mathbf{v}}] = \frac{1}{(1-2p)} \sum_{j=1}^{|C|} (\mathbb{E}[y_j] - p) \mathbf{c}_j, \quad (3.19)$$

where, the expectation is taken over the randomness of both the quantization and RAPPOR scheme.

Recall that

$$\mathbf{y} := \text{RAPPOR}_p(1\text{-HOT}(Q_C(\mathbf{v}), C)) \in \{0, 1\}^{|C|}.$$

Each entry of the vector \mathbf{y} is an independent binary random variable and

$$\begin{aligned} \mathbb{E}[y_j] &= \Pr(y_j = 1) = \sum_{i=1}^{|C|} \Pr(y_j, Q_C(\mathbf{v}) = \mathbf{c}_i) \\ &= \sum_{i=1}^{|C|} \Pr(y_j | Q_C(\mathbf{v}) = \mathbf{c}_i) \Pr(Q_C(\mathbf{v}) = \mathbf{c}_i) \\ &= \Pr(y_j | Q_C(\mathbf{v}) = \mathbf{c}_j) \Pr(Q_C(\mathbf{v}) = \mathbf{c}_j) \\ &\quad + \sum_{i \neq j}^{|C|} \Pr(y_j | Q_C(\mathbf{v}) = \mathbf{c}_i) \Pr(Q_C(\mathbf{v}) = \mathbf{c}_i) \\ &= (1-p)a_j + p(1-a_j) = p + (1-2p)a_j. \end{aligned} \quad (3.20)$$

Plugging Equation (3.20) in Equation (3.19), we get

$$\mathbb{E}(\hat{\mathbf{v}}) = \frac{1}{(1-2p)} \sum_{j=1}^{|C|} (p + (1-2p)a_j - p) \mathbf{c}_j = \sum_{j=1}^{|C|} a_j \mathbf{c}_j = \mathbf{v} \quad (3.21)$$

Now we show a bound on the variance of the estimate

$$\mathbb{E} [\|\mathbf{v} - \hat{\mathbf{v}}\|_2^2] = \mathbb{E} \left[\left\| \sum_{j=1}^{|C|} a_j \mathbf{c}_j - \frac{1}{(1-2p)} \sum_{j=1}^{|C|} (y_j - p) \mathbf{c}_j \right\|_2^2 \right] \quad (3.22)$$

$$= \sum_{j=1}^{|C|} \mathbb{E} \left(a_j - \frac{(y_j - p)}{(1-2p)} \right)^2 |\mathbf{c}_j|^2 \quad (3.23)$$

(all the cross terms are 0 as they are mutually independent and $\mathbb{E} \left(a_j - \frac{y_j - p}{1-2p} \right) = 0$)

$$= \sum_{j=1}^{|C|} \text{var} \left(\frac{y_j - p}{1-2p} \right) |\mathbf{c}_j|^2 \quad (3.24)$$

$$= \left(\frac{1}{1-2p} \right)^2 \sum_{j=1}^{|C|} \text{var}(y_j) |\mathbf{c}_j|^2 = O(|C|R^2) \quad (3.25)$$

Equation (3.25) comes from the fact that y_j is a binary random variable and $\text{Var}(y_j) = \text{Pr}(y_j)(1 - \text{Pr}(y_j)) \leq \frac{1}{4}$ and $|\mathbf{c}_j|^2 \leq R^2$.

Privacy Now we show that our scheme is ϵ differentially private where ϵ is the input parameter to the RAPPOR algorithm. For any two points $\mathbf{v}, \mathbf{w} \in B_d(\mathbf{0}_d, 1)$,

$$\frac{\text{Pr}(Q_{C,\epsilon}(\mathbf{v}) = y)}{\text{Pr}(Q_{C,\epsilon}(\mathbf{w}) = y)} = \frac{\sum_{i=1}^{|C|} \text{Pr}(y|Q_C(\mathbf{v}) = \mathbf{c}_i) \text{Pr}(Q_C(\mathbf{v}) = \mathbf{c}_i)}{\sum_{j=1}^{|C|} \text{Pr}(y|Q_C(\mathbf{w}) = \mathbf{c}_j) \text{Pr}(Q_C(\mathbf{w}) = \mathbf{c}_j)} \quad (3.26)$$

$$\leq \frac{\max_i \text{Pr}(y|Q_C(\mathbf{v}) = \mathbf{c}_i) \sum_{i=1}^{|C|} \text{Pr}(Q_C(\mathbf{v}) = \mathbf{c}_i)}{\min_j \text{Pr}(y|Q_C(\mathbf{w}) = \mathbf{c}_j) \sum_{j=1}^{|C|} \text{Pr}(Q_C(\mathbf{w}) = \mathbf{c}_j)} \quad (3.27)$$

$$= \frac{\max_i \text{Pr}(y|Q_C(\mathbf{v}) = \mathbf{c}_i)}{\min_j \text{Pr}(y|Q_C(\mathbf{w}) = \mathbf{c}_j)} \leq e^\epsilon \quad (3.28)$$

By the privacy property of RAPPOR [45] mechanism , we are using the following fact in equation (3.28)

$$\sup_{i,j} \frac{\text{Pr}(y|Q_C(\mathbf{v}) = \mathbf{c}_i)}{\text{Pr}(y|Q_C(\mathbf{w}) = \mathbf{c}_j)} \leq e^\epsilon \quad \forall \mathbf{v}, \mathbf{w}$$

Communication : Now we show that for the RAPPOR based scheme the expected communication is linear in $|C|$. Say y is the output when RAPPOR is applied to one hot encoded binary string. Without loss of generality say the bit string is \mathbf{e}_i . The output y is generated as follows

$$\Pr(y_j = 1) = \begin{cases} p & \text{if } j \neq i \\ (1 - p) & \text{if } j = i \end{cases}$$

So the expected sparsity (l_0 norm) of the output is

$$\begin{aligned} \mathbb{E}[\|y\|_0] &= \sum_i^{|C|} y_i = (|C| - 1)p + (1 - p) \\ &= |C|p + (1 - 2p) = O(|C|) \end{aligned}$$

□

3.6 Experiments

We use our gradient quantization scheme to train a fully connected ReLU activated network with 1000 hidden nodes using the MNIST [81] and the Fashion MNIST [?] dataset (60000 data points with 10 classes for each). We use the *cross-entropy loss* function for the training the neural network with a total of $d = 795010$ parameters.

The dataset is divided equally among 100 workers. Each worker computes the local gradients and communicates the quantized gradient to the master which then aggregates and send the updated parameters. We plot the error at each iteration (Figure 3.1) and compare our results with QSGD quantization.

We use vqSGD with cross polytope scheme, $Q_{C_{cp}}$, along with the variance reduction technique with repetition parameter $s = 100$. Therefore, each local machine sends about $2060 = 100 \cdot \log(2d)$ bits per iteration whereas, QSGD requires 3825.05 bits

for MNIST and 2266.79 bits for Fashion MNIST, of communication per iteration per machine (computed by averaging over the total bits of communication over 50 iterations) to communicate the quantized gradient. Our results indicate that vqSGD converges at a similar rate to QSGD while communicating much lesser bits.

We also our vqSGD with the cross polytope scheme, Q_{Cp} , to train a ReLU network with 4000 hidden nodes using the CIFAR 10 dataset [78]. This dataset also has 10 classes, every other set up is same except now we have $d = 12332010$ parameters.

The dataset is again equally divided among 100 users. Using vqSGD, each machine send 2455 bits per iteration using the variance reduction scheme. On the other hand, for QSGD, the number of bits per machine per iteration is 4096.9 (computed by averaging over the total bits of communication over 50 iterations). As is evident from the plot in Figure 3.1, vqSGD communicates lesser number of bits to achieve similar performance.

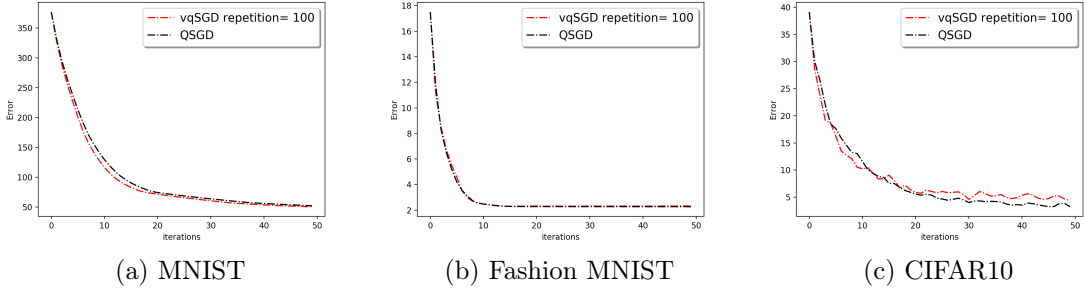


Figure 3.1: Convergence for fully connected ReLU network compared to QSGD

Further we experimentally show the performance of vqSGD using the cross polytope Q_{cp} , to solve the least squares problem and logistic regression for binary classification.

Least Squares: In the least square problem, we solve for $\theta^* = \arg \min_{\theta} \|A\theta - \mathbf{b}\|_2^2$, where the matrix $A \in \mathbb{R}^{n \times d}$ and $\theta^* \in \mathbb{R}^d$ are generated by sampling each entry from $\mathcal{N}(0, 1)$ and we set $\mathbf{b} = A\theta^*$.

In order to show the performance of vqSGD, we simulate the iterations of distributed SGD with $n = 10000$ data samples distributed equally among $N = 500$ worker nodes.

In every iteration of SGD, each worker node computes the local gradient on individual data batch and communicates the quantized version of the local gradient to the parameter server. The parameter server on receiving all the quantized gradients averages them and broadcasts the updated model to all the workers. The convergence of SGD is measured by the error term $\|\boldsymbol{\theta}^* - \boldsymbol{\theta}_t\|_2$, where $\boldsymbol{\theta}_t$ is the computed parameter at the end of t -th iteration of distributed SGD.

We compare the convergence of the least square problem for $d = 100, 200, 500$ against the state-of-the-art quantization schemes - DME [118] and QSGD [6]. The results are presented in Figure 3.2.

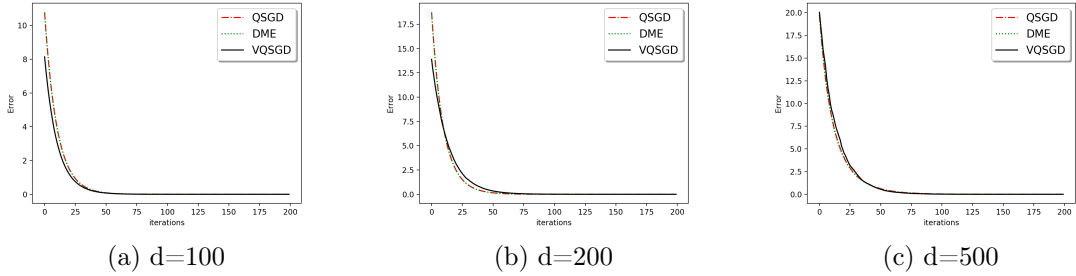


Figure 3.2: Comparison of convergence for the least square problem with $d = 100, 200, 500$.

The results indicate that vqSGD achieves the same rate of convergence and accuracy as DME and QSGD while communicating only $\log(2d)$ bits and one real (l_2 norm of the vector form each server), whereas, DME (one bit stochastic quantization) and QSGD both require communication of about \sqrt{d} bits and one real.

For the same problem setup, we also show the improvement in the performance of vqSGD using the repetition technique for variance reduction. Recall that using repetition technique, each worker now sends s different indices instead of 1 which increases the communication to $s \log(2d)$ bits and 1 real. In Figure 3.3 we plot the convergence of the least square problem with $d = 200$ with different values of $s = 1, 5, 10, 20$. We see the evident improvement in the convergence of vqSGD using this repetition scheme with increasing s .

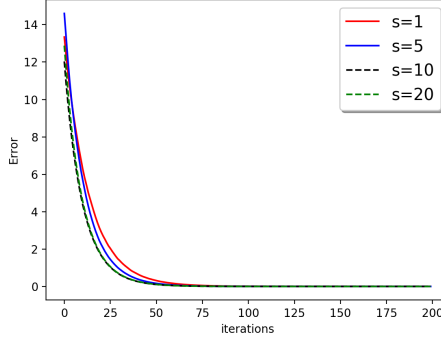


Figure 3.3: Convergence of θ_t for $s = 1, 5, 10, 20$. for least square problem

Binary Classification: We compared the performance of vqSGD against DME and QSGD for the binary classification problem with logistic regression using various datasets from the UCI repository [22]. The logistic regression objective is defined as

$$\frac{1}{n} \sum_{i=1}^n \log(1 + \exp(-b_i \mathbf{a}_i^T \boldsymbol{\theta})) + \frac{1}{2n} \|\boldsymbol{\theta}\|_2^2, \quad (3.29)$$

where $\boldsymbol{\theta} \in \mathbb{R}^d$ is the parameter, $\mathbf{a}_i \in \mathbb{R}^d$ is the feature data and $b_i \in \{-1, +1\}$ is its corresponding label.

We partition the data into 20 equal-sized batches, each assigned to a different worker node. We calculate the classification error for different (test) datasets after training the parameter in the distributed settings (same as described in least square problem). Results of the experiments are presented in Table 3.3, where each entry is averaged over 20 different runs.

Method	DME	QSGD	vqSGD
a5a ($d = 122$)	0.238 ± 0.0003	0.238 ± 0.0002	0.2368 ± 0.0029
a9a ($d = 123$)	0.234 ± 0.0003	0.234 ± 0.00017	0.234 ± 0.0015
gisette-scale ($d = 5000$)	0.0947 ± 0.00384	0.10475 ± 0.006	0.1480 ± 0.0174
splice ($d = 60$)	0.467 ± 0.017	0.4505 ± 0.0352	0.16618 ± 0.0054

Table 3.3: Comparison in classification error (mean \pm standard deviation) for various UCI datasets

We note that for most datasets, with the exception of gisset-scale, vqSGD with $O(N \log d)$ bits of communication per iteration performs equally well or sometimes even better than QSGD and DME with $O(Nd)$ bits of communication per iteration.

3.7 Conclusion and Future Direction

In this chapter, we propose a general framework of convex-hull based private quantization schemes for distributed stochastic gradient descent that can be instantiated with any point set satisfying certain properties. We also provide randomized and deterministic point sets for the quantization and based on the point sets, we achieve different communication, variance and privacy trade-off. With the Gaussian point set, we theoretically establish the communication- variance trade-off and for practical purpose, the repetition scheme is suitable to tune the communication-variance trade-off.

In the future, it would be interesting to consider communication, variance and privacy together. It is to be noted that our cross-polytope scheme which is very efficient and easy to implement, achieve $O(\log d)$ privacy due to the construction while enjoying $O(d)$ variance. But if we want to design quantization for any value of ϵ , in this chapter we used randomized response on top of it. Due to this, the variance blows up. It would be challenging to study whether there is a quantization scheme that can achieve a certain level of privacy, with a communication budget and low variance. In a recent work [25], the authors proposed a deterministic construction with *Kashin representation* to answer this particular question. But a more general theoretical study is warranted to understand the relationship between the iteration complexity of optimization and communication cost under data privacy. Also, in this chapter we considered pure differential privacy with $\delta = 0$. It would be also be interesting to see if a better (lower) value of ϵ can be attained by leveraging some $\delta > 0$.

CHAPTER 4

COMMUNICATION-EFFICIENT AND BYZANTINE-ROBUST DISTRIBUTED LEARNING WITH ERROR FEEDBACK

In this chapter, we develop a communication-efficient distributed statistical learning algorithm that is provably robust against Byzantine workers. Specifically, we consider the following setup. There are m worker machines, each storing n data points. The data points are generated from some unknown distribution \mathcal{D} . The objective is to learn a parametric model that minimizes a population loss function $F : \mathcal{W} \rightarrow \mathbb{R}$, where F is defined as an expectation over \mathcal{D} , and $\mathcal{W} \subseteq \mathbb{R}^d$ denotes the parameter space. We choose the loss function F to be non-convex. With the rapid rise of the neural networks, the study of *local minima* in non-convex optimization framework has become imperative [112, 51]. For gradient compression at workers, we consider the notion of a δ -approximate compressor from [71] that encompasses sign-based compressors like *signSGD* [13] and top- k sparsification [113]. We also assume that $0 \leq \alpha < 1/2$ fraction of the worker machines are Byzantine. We first consider a *restricted* (as described next) adversarial model in Section 4.4, and in the subsequent section, we remove this restriction by slightly changing the learning algorithm.

Our key idea is to use a simple threshold (on local gradient norms) based Byzantine resilience scheme instead of robust aggregation methods such as coordinate wise median or trimmed mean of [135]. We mention that similar ideas are used in *gradient clipping*, where gradients with norm more than a threshold is truncated. This is used in applications like training neural nets [97] to handle the issue of exploding gradients, and in differentially private SGD [26], to limit the sensitivity of the gradients. Note

that although gradient clipping and norm based thresholding have some similarities, they are not identical. In gradient clipping, although we scale down (clip) the gradients, we retain them. On the other hand, in norm based thresholding, we aim to identify the Byzantine machines and remove them. Note that in our learning framework, we have α fraction of Byzantine workers, and an estimate of α is known to the learning algorithm. When α is very close to 0, our learning algorithm does not trim worker machines, and the effect of all gradients are considered. If we employ gradient clipping in this regime, depending on the threshold used in the clipping operation, some gradients may be scaled back. As a result, the convergence rate will suffer. On the other hand, suppose α is large. In this regime, our algorithm tend to identify and remove the influence of the Byzantine workers, where gradient clipping would scale them down, but retain term in the learning process. This could potentially slow down the learning as the Byzantine machines may send *any arbitrary* updates, which are different for the actual gradient norms and directions. Hence, in both the regimes, the knowledge of α helps our algorithm to handle the Byzantine workers gracefully compared to the gradient clipping operation.

Our main result is to show that, for a wide range of compression factor δ , the statistical error rate of our proposed threshold-based scheme is (order-wise) identical to the case of no compression considered in [135]. In fact, our algorithm achieves order-wise optimal error-rate in parameters (α, n, m) . Furthermore, to alleviate convergence issues associated with sign-based compressors, we employ the technique of error-feedback from [71]. In this setup, the worker machines store the difference between the actual and compressed gradient and add it back to the next step so that the *correct* direction of the gradient is not forgotten. We show that using error feedback with our threshold based Byzantine resilience scheme not only achieves better statistical error rate but also improves the rate of convergence. We outline our specific contributions in the following.

Our Contributions: We propose a communication-efficient and robust distributed gradient descent (GD) algorithm. The algorithm takes as input the gradients compressed using a δ -approximate compressor along with the norms¹ (of either compressed or uncompressed gradients), and performs a simple thresholding operation based on gradient norms to discard $\beta > \alpha$ fraction of workers with the largest norm values. We establish the statistical error rate of the algorithm for arbitrary smooth population loss functions as a function of the number of worker machines m , the number of data points on each machine n , dimension d , and the compression factor δ . In particular, we show that our algorithm achieves the following statistical error rate for the regime $\delta > 4\beta + 4\alpha - 8\alpha^2 + 4\alpha^3$:

$$\tilde{\mathcal{O}}\left(d^2 \left[\frac{\alpha^2}{n} + \frac{1-\delta}{n} + \frac{1}{mn} \right] \right). \quad (4.1)$$

We first note that when $\delta = 1$ (uncompressed), the error rate is $\tilde{\mathcal{O}}(d^2[\frac{\alpha^2}{n} + \frac{1}{mn}])$, which matches [135]. Notice that we use a simple threshold (on local gradient norms) based Byzantine resilience scheme in contrast with the coordinate wise median or trimmed mean of [135]. We note that for a fixed d and the compression factor δ satisfying $\delta \geq 1 - \alpha^2$, the statistical error rate become $\tilde{\mathcal{O}}(\frac{\alpha^2}{n} + \frac{1}{mn})$, which is order-wise identical to the case of no compression [135]. In other words, in this parameter regime, the compression term does not contribute (order-wise) to the statistical error. Moreover, it is shown in [135] that, for strongly-convex loss functions and a fixed d , no algorithm can achieve an error lower than $\tilde{\Omega}(\frac{\alpha^2}{n} + \frac{1}{mn})$, implying that our algorithm is order-wise optimal in terms of the statistical error rate in the parameters (α, n, m) .

Furthermore, we strengthen our distributed learning algorithm by using error feedback to correct the direction of the local gradient. We show (both theoretically and via experiments) that using error-feedback with a δ -approximate compressor indeed

¹We can handle any convex norm.

speeds up the convergence rate and attains better (statistical) error rate. Under the assumption that the gradient norm of the local loss function is upper-bounded by σ , we obtain the following (statistical) error rate:

$$\tilde{\mathcal{O}}\left(d^2 \left[\frac{\alpha^2}{n} + \frac{(1-\delta)\sigma^2}{d^2 \delta} + \frac{1}{mn} \right]\right)$$

provided a similar (δ, α) trade-off². We note that in the no-compression setting ($\delta = 1$), we recover the $\tilde{\mathcal{O}}(\frac{\alpha^2}{n} + \frac{1}{mn})$ rate. In experiments (Section ??), we see that adding error feedback indeed improves the performance of our algorithm.

We experimentally evaluate our algorithm for convex and non-convex losses. For the convex case, we choose the linear regression problem, and for the non-convex case, we train a ReLU activated feed-forward fully connected neural net. We compare our algorithm with the non-Byzantine case and *signSGD* with majority vote, and observe that our algorithm converges faster using the standard MNIST dataset.

A major technical challenge here is to handle compression and the Byzantine behavior of the worker machines simultaneously. We build up on the techniques of [135] to control the Byzantine machines. In particular, using certain distributional assumption on the partial derivative of the loss function and exploiting uniform bounds via careful covering arguments, we show that the local gradient on a non-Byzantine worker machine is close to the gradient of the population loss function.

Organization: We describe the problem formulation in Section 4.1, and give a brief overview of δ -compressors in Section 4.2. Then, we present our proposed algorithm in Section 4.3. We analyze the algorithm, first, for a *restricted* (as described next) adversarial model in Section 4.4, and in the subsequent section, remove this restriction. In Section 4.4, we restrict our attention to an adversarial model in

²See Theorem 4.7 for details.

which Byzantine workers can provide arbitrary values as an input to the compression algorithm, but they correctly implement the same compression scheme as mandated.

In Section 4.5, we remove this restriction on the Byzantine machines. As a consequence, we observe (in Theorem 4.6) that the modified algorithm works under a stricter assumption, and performs slightly worse than the one in restricted adversary setting. In Section 4.6, we strengthen our algorithm by including error-feedback at worker machines, and provide statistical guarantees for non-convex smooth loss functions. We show that error-feedback indeed improves the performance of our optimization algorithm in the presence of arbitrary adversaries.

Related Work

Gradient Compression: The foundation of gradient quantization was laid in [114, 103]. In the work of [6, 131, 125] each co-ordinate of the gradient vector is represented with a small number of bits. Using this, an unbiased estimate of the gradient is computed. In these works, the communication cost is $\Omega(\sqrt{d})$ bits. In [118], a quantization scheme was proposed for distributed mean estimation. The tradeoff between communication and accuracy is studied in [139]. Variance reduction in communication efficient stochastic distributed learning has been studied in [61]. Sparsification techniques are also used instead of quantization to reduce communication cost. Gradient sparsification has been studied in [113, 8, 62] with provable guarantees. The main idea is to communicate top components of the d -dimensional local gradient to get good estimate of the true global gradient.

Byzantine Robust Optimization: In the distributed learning context, a generic framework of one shot median based robust learning has been proposed in [?]. In [27] the issue of Byzantine failure is tackled by grouping the servers in batches and computing the median of batched servers. Later in [135, 136], co-ordinate wise median, trimmed mean and iterative filtering based algorithm have been proposed and

optimal statistical error rate is obtained. Also, [92, 33] considers adversaries may steer convergence to bad local minimizers. In this work, we do not assume such adversaries.

Gradient compression and Byzantine robust optimization have simultaneously been addressed in a recent paper [13]. Here, the authors use *signSGD* as compressor and majority voting as robust aggregator. As explained in [71], *signSGD* can run into convergence issues. Also, [13] can handle a restricted class of adversaries that are *multiplicative* (i.e., multiply each coordinate of gradient by arbitrary scalar) and *blind* (i.e., determine how to corrupt the gradient before observing the true gradient). In this paper, for compression, we use a generic δ approximate compressor. Also, we can handle arbitrary Byzantine worker machines.

Very recently, [71] uses error-feedback to remove some of the issues of sign based compression schemes. In this work, we extend the framework to a distributed setting and prove theoretical guarantees in the presence of Byzantine worker machines.

Throughout the paper, we assume $C, C_1, C_2, \dots, c, c_1, \dots$ as positive universal constants, the value of which may differ from instance to instance.

4.1 Problem Formulation

In this section, we formally set up the problem. We consider a standard statistical problem of risk minimization. In a distributed setting, suppose we have one central and m worker nodes and the worker nodes communicate to the central node. Each worker node contains n data points. We assume that the mn data points are sampled independently from some unknown distribution \mathcal{D} . Also, let $f(\mathbf{w}, \mathbf{z})$ be the non-convex loss function of a parameter vector $\mathbf{w} \in \mathcal{W} \subseteq \mathbb{R}^d$ corresponding to data point \mathbf{z} , where \mathcal{W} is the parameter space. Hence, the population loss function is $F(\mathbf{w}) = \mathbb{E}_{\mathbf{z} \sim \mathcal{D}}[f(\mathbf{w}, \mathbf{z})]$. Our goal is to obtain the following:

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w} \in \mathcal{W}} F(\mathbf{w}),$$

where we assume \mathcal{W} to be a convex and compact subset of \mathbb{R}^d with diameter D . In other words, we have $\|\mathbf{w}_1 - \mathbf{w}_2\| \leq D$ for all $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{W}$. Each worker node is associated with a local loss defined as $F_i(\mathbf{w}) = \frac{1}{n} \sum_{j=1}^n f(\mathbf{w}, \mathbf{z}^{i,j})$, where $\mathbf{z}^{i,j}$ denotes the j -th data point in the i -th machine. This is precisely the empirical risk function of the i -th worker node.

We assume a setup where worker i compresses the local gradient and sends to the central machine. The central machine aggregates the compressed gradients, takes a gradient step to update the model and broadcasts the updated model to be used in the subsequent iteration. Furthermore, we assume that α fraction of the total workers nodes are Byzantine, for some $\alpha < 1/2$. Byzantine workers can send any arbitrary values to the central machine. In addition, Byzantine workers may completely know the learning algorithm and are allowed to collude with each other.

Next, we define a few (standard) quantities that will be required in our analysis.

Definition 4.1. (*Sub-exponential random variable*) A zero mean random variable Y is called v -sub-exponential if $\mathbb{E}[e^{\lambda Y}] \leq e^{\frac{1}{2}\lambda^2 v^2}$, for all $|\lambda| \leq \frac{1}{v}$.

Definition 4.2. (*Smoothness*) A function $h(\cdot)$ is L_F -smooth if $h(\mathbf{w}) \leq h(\mathbf{w}') + \langle \nabla h(\mathbf{w}'), \mathbf{w} - \mathbf{w}' \rangle + \frac{L_F}{2} \|\mathbf{w} - \mathbf{w}'\|^2 \forall \mathbf{w}, \mathbf{w}'$.

Definition 4.3. (*Lipschitz*) A function $h(\cdot)$ is L -Lipschitz if $\|h(\mathbf{w}) - h(\mathbf{w}')\| \leq L\|\mathbf{w} - \mathbf{w}'\| \forall \mathbf{w}, \mathbf{w}'$.

4.2 Compression at Worker Machines

In this section, we consider a generic class of compressors from [113] and [71] as described in the following.

Algorithm 1 Robust Compressed Gradient Descent

1: **Input:** Step size γ , Compressor $\mathcal{Q}(\cdot)$, $q > 1$, $\beta < 1$. Also define,

$$\mathcal{C}(\mathbf{x}) = \begin{cases} \{\mathcal{Q}(\mathbf{x}), \|\mathbf{x}\|_q\} & \forall \mathbf{x} \in \mathbb{R}^d \quad \text{Option I} \\ \mathcal{Q}(\mathbf{x}) & \forall \mathbf{x} \in \mathbb{R}^d \quad \text{Option II} \end{cases}$$

2: **Initialize:** Initial iterate \mathbf{w}_0

3: **for** $t = 0, 1, \dots, T - 1$ **do**

4: Central machine: broadcasts \mathbf{w}_t

for $i \in [m]$ **do in parallel**

5: i -th worker machine:

- (Non-Byzantine) computes $\nabla F_i(\mathbf{w}_t)$; sends $\mathcal{C}(\nabla F_i(\mathbf{w}_t))$ to the central machine
- (Byzantine) generates \star (arbitrary), and sends $\mathcal{C}(\star)$ to the central machine

end for

6: Central Machine:

- Sort the local gradient norms in a non decreasing order
- Return the indices of the first $1 - \beta$ fraction of elements as \mathcal{U}_t .
- Update model parameter: $\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\gamma}{|\mathcal{U}_t|} \sum_{i \in \mathcal{U}_t} \mathcal{Q}(\nabla F_i(\mathbf{w}_t))$

7: **end for**

Definition 4.4 (δ -Approximate Compressor). *An operator $\mathcal{Q}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is defined as δ -approximate compressor on a set $\mathcal{S} \subseteq \mathbb{R}^d$ if, $\forall \mathbf{x} \in \mathcal{S}$,*

$$\|\mathcal{Q}(\mathbf{x}) - \mathbf{x}\|^2 \leq (1 - \delta)\|\mathbf{x}\|^2,$$

where $\delta \in (0, 1]$ is the compression factor.

Furthermore, a randomized operator $\mathcal{Q}(\cdot)$ is δ -approximate compressor on a set $\mathcal{S} \subseteq \mathbb{R}^d$ if,

$$\mathbb{E} (\|\mathcal{Q}(\mathbf{x}) - \mathbf{x}\|^2) \leq (1 - \delta)\|\mathbf{x}\|^2$$

holds for all $\mathbf{x} \in \mathcal{S}$, where the expectation is taken with respect to the randomness of $\mathcal{Q}(\cdot)$. For the clarity of exposition, we consider the deterministic form of the compressor (as in Definition 4.4). However, the results can be easily extended for randomized $\mathcal{Q}(\cdot)$.

Notice that $\delta = 1$ implies $\mathcal{Q}(\mathbf{x}) = \mathbf{x}$ (no compression). We list a few examples of δ -approximate compressors (including a few from [71]) here:

1. top_k operator, which selects k coordinates with largest absolute value; for $1 \leq k \leq d$, $(\mathcal{Q}(\mathbf{x}))_i = (\mathbf{x})_{\pi(i)}$ if $i \leq k$, and 0 otherwise, where π is a permutation of $[d]$ with $(|\mathbf{x}|)_{\pi(i)} \geq (|\mathbf{x}|)_{\pi(i+1)}$ for $i \in [d-1]$. This is a k/d -approximate compressor.
2. k -PCA that uses top k eigenvectors to approximate a matrix \mathbf{X} ([125]).
3. Quantized SGD (QSGD) [6], where $\mathcal{Q}(x_i) = \|\mathbf{x}\| \cdot \text{sign}(x_i) \cdot \xi_i(x)$, where $\text{sign}(x_i)$ is the coordinate-wise sign vector, and $\xi_i(x)$ is defined as following: let $0 \leq l \leq s$, be an integer such that $|x_i|/\|\mathbf{x}\| \in [l/s, (l+1)/s]$. Then, $\xi_i = l/s$ with probability $1 - \frac{|x_i|}{c\|\mathbf{x}\|\sqrt{d}} + l$ and $(l+1)/s$ otherwise. [6] shows that it is a $1 - \min(d/s^2, \sqrt{d}/s)$ -approximate compressor.
4. Quantized SGD with ℓ_1 norm [71], $\mathcal{Q}(\mathbf{x}) = \frac{\|\mathbf{x}\|_1}{d} \text{sign}(\mathbf{x})$, which is $\frac{\|\mathbf{x}\|_1^2}{d\|\mathbf{x}\|^2}$ -approximate compressor. In this paper, we call this compression scheme as ℓ_1 -QSGD.

Apart from these examples, several randomized compressors are also discussed in [113]. Also, the *signSGD* compressor, $\mathcal{Q}(\mathbf{x}) = \text{sign}(\mathbf{x})$, where $\text{sign}(\mathbf{x})$ is the (coordinate-wise) sign operator, was proposed in [11, 12]. Here the local machines send a d -dimensional vector containing coordinate-wise sign of the gradients.

4.3 Robust Compressed Gradient Descent

In this section, we describe a communication-efficient and robust distributed gradient descent algorithm for δ -approximate compressors. The optimization algorithm

we use is formally given in Algorithm 1. Note that the algorithm uses a compression scheme $\mathcal{Q}(\cdot)$ to reduce communication cost and a norm based thresholding to remove Byzantine worker nodes. The idea behind norm based thresholding is quite intuitive. Note that, if the Byzantine worker machines try to diverge the learning algorithm by increasing the norm of the local gradients; Algorithm 1 can identify them as outliers. Furthermore, when the Byzantine machines behave like inliers, they can not diverge the learning algorithm since they are only a few ($\alpha < 1/2$) in number. It turns out that this simple approach indeed works.

As seen in Algorithm 1, robust compressed gradient descent operates under two different settings, namely *Option I* and *Option II*. Option I and II are analyzed in Sections 4.4 and 4.5 respectively. For Option I, we use a δ -approximate compressor along with the norm information. In particular, the worker machines send the pair denoted by $\mathcal{C}(\mathbf{x}) = \{\mathcal{Q}(\mathbf{x}), \|\mathbf{x}\|_q\}$ where we have $q \geq 1$, to the center machine. $\mathcal{C}(\mathbf{x})$ is comprised of a scalar (norm of \mathbf{x}) and a compressed vector $\mathcal{Q}(\mathbf{x})$. For compressors such as QSGD ([6]) and ℓ_1 -QSGD ([71]), the quantity $\mathcal{Q}(\cdot)$ has the norm information and hence sending the norm separately is not required.

As seen in Option I of Algorithm 4, worker node i compresses the local gradient $\nabla F_i(\cdot)$ and sends $\mathcal{C}(\nabla F_i(\cdot))$ to the central machine. Adversary nodes can send arbitrary $\mathcal{C}(\star)$ to the central machine. The central machine aggregates the gradients, takes a gradient step and broadcasts the updated model for next iteration.

For Option I, we restrict to the setting where the Byzantine worker machines can send arbitrary values to the input of the compression algorithm, but they adhere to the compression algorithm. In particular, Byzantine workers can provide arbitrary values, \star to the input of the compression algorithm, $\mathcal{Q}(\cdot)$ but they correctly implement the same compression algorithm, i.e., computes $\mathcal{Q}(\star)$.

We now explain how Algorithm 1 tackles the Byzantine worker machines. The central machine receives the compressed gradients comprising a scalar ($\|\mathbf{x}\|_q, q \geq 1$)

and a quantized vector ($\mathcal{Q}(\mathbf{x})$) and outputs a set of indices \mathcal{U} with $|\mathcal{U}| = (1 - \beta)m$. Here we employ a simple thresholding scheme on the (local) gradient norm.

Note that, if the Byzantine worker machines try to diverge the learning algorithm by increasing the norm of the local gradients; Algorithm 1 can identify them as outliers. Furthermore, when the Byzantine machines behave like inliers, they can not diverge the learning algorithm since $\alpha < 1/2$. In the subsequent sections, we show theoretical justification of this argument.

With Option II, we remove this restriction on Byzantine machines at the cost of slightly weakening the convergence guarantees. This is explained in Section 4.5. With Option II, the i -th local machine sends $\mathcal{C} = \{\mathcal{Q}(\nabla F_i(\mathbf{w}_t)), \|\mathcal{Q}(\nabla F_i(\mathbf{w}_t))\|_q\}$ to the central machine, where $q \geq 1$. Effectively, the i -th local machine just sends $\mathcal{Q}(\nabla F_i(\mathbf{w}_t))$ since its norm can be computed at the central machine. Byzantine workers just send arbitrary (\star) vector instead of compressed local gradient. Note that the Byzantine workers here do not adhere to any compression rule.

The Byzantine resilience scheme with Option II is similar to Option I except the fact that the central machine sorts the worker machines according to the norm of the compressed gradients rather than the norm of the gradients.

4.4 Distributed Learning with Restricted Adversaries

In this section, we analyze the performance of Algorithm 1 with *Option I*. We restrict to an adversarial model in which Byzantine workers can provide arbitrary values to the input of the compression algorithm, but they adhere to the compression rule. Though this adversarial model is restricted, we argue that it is well-suited for applications wherein compression happens outside of worker machines. For example, Apache MXNet, a deep learning framework designed to be distributed on cloud infrastructures, uses NVIDIA Collective Communication Library (NCCL) that employs gradient compression (see [1]). Also, in a Federated Learning setup the compression can be part

of the communication protocol. Furthermore, this can happen when worker machines are divided into groups, and each group is associated with a compression unit. As an example, cores in a multi-core processor ([85]) acting as a group of worker machines with the compression carried out by a separate processor, or servers co-located on a rack ([31]) acting as a group with the compression carried out by the top-of-the-rack switch.

4.4.1 Main Results

We analyze Algorithm 1 (with Option I) and obtain the rate of the convergence under non-convex loss functions. We start with the following assumption.

Assumption 4.1. *For all \mathbf{z} , the partial derivative of the loss function $f(., \mathbf{z})$ with respect to the k -th coordinate (denoted as $\partial_k f(., \mathbf{z})$) is L_k Lipschitz with respect to the first argument for each $k \in [d]$, and let $\widehat{L} = \sqrt{\sum_{i=1}^d L_k^2}$. The population loss function $F(.)$ is L_F smooth.*

We also make the following assumption on the tail behavior of the partial derivative of the loss function.

Assumption 4.2. *(Sub-exponential gradients) For all $k \in [d]$ and \mathbf{z} , the quantity $\partial_k f(\mathbf{w}, \mathbf{z})$ is v sub-exponential for all $\mathbf{w} \in \mathcal{W}$.*

The assumption implies that the moments of the partial derivatives are bounded. We like to emphasize that the sub-exponential assumption on gradients is fairly common ([135, 115, 132]). For instance, [135, Proposition 2] gives a concrete example of coordinate-wise sub-exponential gradients in the context of a regression problem. Furthermore, in [136], the gradients are assumed to be sub-gaussian, which is stronger than Assumption 4.2.

To simplify notation and for the clarity of exposition, we define the following three quantities which will be used throughout the paper.

$$\epsilon_1 = v\sqrt{d} \left(\max \left\{ \frac{d}{n} \log(1 + 2nD\hat{L}d), \sqrt{\frac{d}{n} \log(1 + 2nD\hat{L}d)} \right\} \right) + \frac{1}{n}, \quad (4.2)$$

$$\epsilon_2 = v\sqrt{d} \left(\max \left\{ \frac{d}{(1-\alpha)mn} \log(1 + 2(1-\alpha)mnD\hat{L}d), \sqrt{\frac{d}{(1-\alpha)mn} \log(1 + 2(1-\alpha)mnD\hat{L}d)} \right\} \right), \quad (4.3)$$

$$\epsilon = 2 \left(1 + \frac{1}{\lambda_0} \right) \left[\left(\frac{1-\alpha}{1-\beta} \right)^2 \epsilon_2^2 + \left(\frac{\sqrt{1-\delta} + \alpha + \beta}{1-\beta} \right)^2 \epsilon_1^2 \right]. \quad (4.4)$$

where λ_0 is a positive constant. For intuition, one can think of $\epsilon_1 = \tilde{\mathcal{O}}(\frac{d}{\sqrt{n}})$ and $\epsilon_2 = \tilde{\mathcal{O}}(\frac{d}{\sqrt{mn}})$ as small problem dependent quantities. Assuming $\beta = c\alpha$ for a universal constant $c > 1$, we have

$$\epsilon = \tilde{\mathcal{O}} \left(d^2 \left[\frac{\alpha^2}{n} + \frac{1-\delta}{n} + \frac{1}{mn} \right] \right). \quad (4.5)$$

Assumption 4.3. (*Size of parameter space \mathcal{W}*) Suppose that $\|\nabla F(\mathbf{w})\| \leq M$ for all $\mathbf{w} \in \mathcal{W}$. We assume that \mathcal{W} contains the ℓ_2 ball $\{\mathbf{w} : \|\mathbf{w} - \mathbf{w}_0\| \leq c[(2 - \frac{c_0}{2})M + \sqrt{\epsilon}] \frac{F(\mathbf{w}_0) - F(\mathbf{w}^*)}{\epsilon}\}$, where c_0 is a constant, δ is the compression factor, \mathbf{w}_0 is the initial parameter vector and ϵ is defined in equation (4.4).

We use the above assumption to ensure that the iterates of Algorithm 1 stays in \mathcal{W} . We emphasize that this is a standard assumption on the size of \mathcal{W} to control the iterates for non-convex loss function. Note that, similar assumptions have been used in prior works [135, Assumption 5], [136]. We point out that Assumption 4.3 is used for simplicity and is not a hard requirement. We show (in the proof of Theorem 4.4) that the iterates of Algorithm 1 stay in a bounded set around the initial iterate \mathbf{w}_0 . Also, note that the dependence of M in the final statistical rate (implicit, via diameter D) is logarithmic (weak dependence), as will be seen in Theorem 4.4.

First, we make the state the following fact. Let \mathcal{M} and \mathcal{B} denote the set of non-Byzantine and Byzantine worker machines. Furthermore, \mathcal{U}_t and \mathcal{T}_t denote untrimmed and trimmed worker machines. So evidently,

$$|\mathcal{M}| + |\mathcal{B}| = |\mathcal{U}_t| + |\mathcal{T}_t| = m.$$

We provide the following rate of convergence to a critical point of the (non-convex) population loss function $F(\cdot)$. In the following result, we show that for non-Byzantine worker machine i , the local gradient $\nabla F_i(\mathbf{w}_t)$ is concentrated around the global gradient $\nabla F(\mathbf{w}_t)$.

Lemma 4.1. *For any $w \in \mathcal{W}$, we have*

$$\max_{i \in \mathcal{M}} \|\nabla F_i(\mathbf{w}) - \nabla F(\mathbf{w})\| \leq \epsilon_1$$

with probability exceeding $1 - \frac{2(1-\alpha)md}{(1+nLD)^d}$, where ϵ_1 is defined in equation (4.2).

Proof. Proof of Lemma 4.1: For a fixed $i \in \mathcal{M}$, we first analyze the quantity $\|\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)\|$. Notice that i is non-Byzantine. Recall that machine i has n independent data points. We use the sub-exponential concentration to control this term. Let us rewrite the concentration inequality.

Univariate sub-exponential concentration: Suppose Y is univariate random variable with $\mathbb{E}Y = \mu$ and y_1, \dots, y_n are i.i.d draws of Y . Also, Y is v sub-exponential. From sub-exponential concentration (Hoeffding's inequality), we obtain

$$\Pr \left(\left| \frac{1}{n} \sum_{i=1}^n y_i - \mu \right| > t \right) \leq 2 \exp \left\{ -n \min \left(\frac{t}{v}, \frac{t^2}{v^2} \right) \right\}.$$

We directly use this to the k -th partial derivative of F_i . Let $\partial_k f(\mathbf{w}_t, \mathbf{z}^{i,j})$ be the partial derivative of the loss function with respect to k -th coordinate on i -th machine with j -th data point. From Assumption 4.2, we obtain

$$\Pr \left(\left| \frac{1}{n} \sum_{j=1}^n \partial_k f(\mathbf{w}_t, \mathbf{z}^{i,j}) - \partial_k F(\mathbf{w}_t) \right| \geq t \right) \leq 2 \exp \left\{ -n \min \left(\frac{t}{v}, \frac{t^2}{v^2} \right) \right\}.$$

Since $\nabla F_i(\mathbf{w}_t) = \frac{1}{n} \sum_{j=1}^n \nabla f(\mathbf{w}_t, \mathbf{z}^{i,j})$, denoting $\nabla F_i^{(k)}(\mathbf{w}_t)$ as the k -th coordinate of $\nabla F_i(\mathbf{w}_t)$, we have

$$|\nabla F_i^{(k)}(\mathbf{w}_t) - \partial_k F(\mathbf{w}_t)| \leq t$$

with probability at least $1 - 2 \exp \left\{ -n \min \left(\frac{t}{v}, \frac{t^2}{v^2} \right) \right\}$.

This result holds for a particular \mathbf{w}_t . To extend this for all $\mathbf{w} \in \mathcal{W}$, we exploit the covering net argument and the Lipschitz continuity of the partial derivative of the loss function (Assumption 4.1). Let $\{\mathbf{w}_1, \dots, \mathbf{w}_N\}$ be a δ covering of \mathcal{W} . Since \mathcal{W} has diameter D , from Vershynin, we obtain $N \leq (1 + \frac{D}{\delta})^d$. Hence with probability at least

$$1 - 2Nd \exp \left\{ -n \min \left(\frac{t}{v}, \frac{t^2}{v^2} \right) \right\},$$

we have

$$|\nabla F_i^{(k)}(\mathbf{w}) - \partial_k F(\mathbf{w})| \leq t$$

for all $\mathbf{w} \in \{\mathbf{w}_1, \dots, \mathbf{w}_N\}$ and $k \in [d]$. This implies

$$\|\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)\| \leq t\sqrt{d},$$

with probability greater than or equal to $1 - 2Nd \exp \left\{ -n \min \left(\frac{t}{v}, \frac{t^2}{v^2} \right) \right\}$.

We now reason about $w \in \mathcal{W} \setminus \{\mathbf{w}_1, \dots, \mathbf{w}_N\}$ via Lipschitzness (Assumption 4.1). From the definition of δ cover, for any $\mathbf{w} \in \mathcal{W}$, there exists \mathbf{w}_ℓ , an element of the cover such that $\|\mathbf{w} - \mathbf{w}_\ell\| \leq \delta$. Hence, we obtain

$$|\nabla F_i^{(k)}(\mathbf{w}) - \partial_k F(\mathbf{w})| \leq t + 2L_k \delta$$

for all $\mathbf{w} \in \mathcal{W}$ and consequently

$$\|\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)\| \leq \sqrt{d} t + 2\delta \hat{L}$$

with probability at least $1 - 2Nd \exp\{-n \min(\frac{t}{v}, \frac{t^2}{v^2})\}$, where $\hat{L} = \sqrt{\sum_{k=1}^d L_k^2}$.

Choosing $\delta = \frac{1}{2n\hat{L}}$ and

$$t = v \max\left\{\frac{d}{n} \log(1 + 2n\hat{L}d), \sqrt{\frac{d}{n} \log(1 + 2n\hat{L}d)}\right\},$$

we obtain

$$\begin{aligned} \|\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)\| &\leq v\sqrt{d} \left(\max\left\{\frac{d}{n} \log(1 + 2n\hat{L}d), \sqrt{\frac{d}{n} \log(1 + 2n\hat{L}d)}\right\} \right) + \frac{1}{n} \\ &= \epsilon_1, \end{aligned} \tag{4.6}$$

with probability greater than $1 - \frac{d}{(1+n\hat{L}D)^d}$. Taking union bound on all non-Byzantine machines yields the lemma. □

Since the iterations $\{w_t\}_{t=1}^T \in \mathcal{W}$, we have the above lemma for all the iterates of our algorithm. Furthermore, we have the following Lemma which implies that the average of local gradients $\nabla F_i(\mathbf{w}_t)$ over non-Byzantine worker machines is close to its expectation $\nabla F(\mathbf{w}_t)$.

Lemma 4.2. *For any $\mathbf{w} \in \mathcal{W}$, we have*

$$\left\| \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \nabla F_i(\mathbf{w}) - \nabla F(\mathbf{w}) \right\| \leq \epsilon_2.$$

with probability exceeding $1 - \frac{2(1-\alpha)md}{(1+n\hat{L}D)^d} - \frac{2d}{(1+(1-\alpha)mn\hat{L}D)^d}$, where ϵ_2 is defined in equation (4.3).

Proof. Proof of Lemma 4.2

We need to upper bound the following quantity:

$$\left\| \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} (\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)) \right\|$$

We now use similar argument (sub-exponential concentration) like Lemma 4.1. The only difference is that in this case, we also consider *averaging* over worker nodes. We obtain the following:

$$\left\| \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} (\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)) \right\| \leq \epsilon_2$$

where

$$\epsilon_2 = v\sqrt{d} \left(\max \left\{ \frac{d}{(1-\alpha)mn} \log(1 + 2(1-\alpha)mn\hat{L}d), \sqrt{\frac{d}{(1-\alpha)mn} \log(1 + 2(1-\alpha)mn\hat{L}d)} \right\} \right),$$

with probability $1 - \frac{2d}{(1+(1-\alpha)mn\hat{L}D)^d}$.

□

Lemma 4.3. For any $\lambda > 0$, we have,

$$\|\Delta\|^2 \leq (1 + \lambda) \left(\frac{\sqrt{1 - \delta} + 2\alpha}{1 - \beta} \right)^2 \|\nabla F(\mathbf{w}_t)\|^2 + \tilde{\epsilon}(\lambda)$$

with probability greater than or equal to $1 - \frac{c_1(1-\alpha)md}{(1+n\hat{L}D)^d} - \frac{c_2d}{(1+(1-\alpha)mn\hat{L}D)^d}$, where

$$\tilde{\epsilon}(\lambda) = 2\left(1 + \frac{1}{\lambda}\right) \left[\left(\frac{\sqrt{1 - \delta} + \alpha + \beta}{1 - \beta} \right)^2 \epsilon_1^2 + \left(\frac{1 - \alpha}{1 - \beta} \right)^2 \epsilon_2^2 \right].$$

with ϵ_1 and ϵ_2 as defined in equation (4.2) and (4.3) respectively.

Proof. Proof of Lemma 4.3: Recall the definition of Δ . Using triangle inequality, we obtain

$$\|\Delta\| \leq \underbrace{\left\| \frac{1}{|\mathcal{U}_t|} \sum_{i \in \mathcal{U}_t} \mathcal{Q}(\nabla F_i(\mathbf{w}_t)) - \frac{1}{|\mathcal{U}_t|} \sum_{i \in \mathcal{U}_t} \nabla F_i(\mathbf{w}_t) \right\|}_{T_1} + \underbrace{\left\| \frac{1}{|\mathcal{U}_t|} \sum_{i \in \mathcal{U}_t} \nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t) \right\|}_{T_2}$$

We first control T_1 . Using the compression scheme (Definition 4.4), we obtain

$$\begin{aligned} T_1 &= \left\| \frac{1}{|\mathcal{U}_t|} \sum_{i \in \mathcal{U}_t} \mathcal{Q}(\nabla F_i(\mathbf{w}_t)) - \frac{1}{|\mathcal{U}_t|} \sum_{i \in \mathcal{U}_t} \nabla F_i(\mathbf{w}_t) \right\| \leq \frac{\sqrt{1 - \delta}}{|\mathcal{U}_t|} \sum_{i \in \mathcal{U}_t} \|\nabla F_i(\mathbf{w}_t)\| \\ &\leq \frac{\sqrt{1 - \delta}}{|\mathcal{U}_t|} \left[\sum_{i \in \mathcal{M}} \|\nabla F_i(\mathbf{w}_t)\| - \sum_{i \in \mathcal{M} \cap \mathcal{T}_t} \|\nabla F_i(\mathbf{w}_t)\| + \sum_{i \in \mathcal{B} \cap \mathcal{U}_t} \|\nabla F_i(\mathbf{w}_t)\| \right] \\ &\leq \frac{\sqrt{1 - \delta}}{|\mathcal{U}_t|} \left[\sum_{i \in \mathcal{M}} \|\nabla F_i(\mathbf{w}_t)\| + \sum_{i \in \mathcal{B} \cap \mathcal{U}_t} \|\nabla F_i(\mathbf{w}_t)\| \right] \end{aligned}$$

Since $\beta \geq \alpha$, we ensure that $\mathcal{M} \cap \mathcal{T}_t \neq \emptyset$. We have,

$$\begin{aligned} T_1 &\leq \frac{\sqrt{1 - \delta}}{|\mathcal{U}_t|} \left[\sum_{i \in \mathcal{M}} \|\nabla F_i(\mathbf{w}_t)\| + \alpha m \max_{i \in \mathcal{M}} \|\nabla F_i(\mathbf{w}_t)\| \right] \\ &\leq \underbrace{\frac{\sqrt{1 - \delta}}{|\mathcal{U}_t|} \left[\sum_{i \in \mathcal{M}} \|\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)\| + \sum_{i \in \mathcal{M}} \|\nabla F(\mathbf{w}_t)\| \right]}_{T_3} \end{aligned}$$

$$+ \underbrace{\frac{\alpha m \sqrt{1-\delta}}{|\mathcal{U}_t|} \max_{i \in \mathcal{M}} [\|\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)\| + \|\nabla F(\mathbf{w}_t)\|]}_{T_4}$$

We now upper-bound T_3 . We have

$$\begin{aligned} T_3 &\leq \frac{\sqrt{1-\delta}|\mathcal{M}|}{|\mathcal{U}_t|} \max_{i \in \mathcal{M}} \|\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)\| + \frac{\sqrt{1-\delta}|\mathcal{M}|}{|\mathcal{U}_t|} \|\nabla F(\mathbf{w}_t)\| \\ &\leq \frac{\sqrt{1-\delta}(1-\alpha)}{(1-\beta)} \max_{i \in \mathcal{M}} \|\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)\| + \frac{\sqrt{1-\delta}(1-\alpha)}{(1-\beta)} \|\nabla F(\mathbf{w}_t)\| \\ &\leq \frac{\sqrt{1-\delta}(1-\alpha)}{(1-\beta)} \epsilon_1 + \frac{\sqrt{1-\delta}(1-\alpha)}{(1-\beta)} \|\nabla F(\mathbf{w}_t)\| \end{aligned}$$

with probability exceeding $1 - \frac{2(1-\alpha)md}{(1+n\hat{L}D)^d}$, where we use Lemma 4.1. Similarly, for T_4 , we have

$$T_4 \leq \frac{\sqrt{1-\delta}\alpha}{1-\beta} \epsilon_1 + \frac{\sqrt{1-\delta}\alpha}{1-\beta} \|\nabla F(\mathbf{w}_t)\|.$$

We now control the terms in T_2 . We obtain the following:

$$\begin{aligned} T_2 &\leq \frac{1}{|\mathcal{U}_t|} \left\| \sum_{i \in \mathcal{U}_t} \nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t) \right\| \\ &\leq \frac{1}{|\mathcal{U}_t|} \left\| \sum_{i \in \mathcal{M}} (\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)) - \sum_{i \in \mathcal{M} \cap \mathcal{T}_t} (\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)) + \sum_{i \in \mathcal{B} \cap \mathcal{T}_t} (\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)) \right\| \\ &\leq \frac{1}{|\mathcal{U}_t|} \left\| \sum_{i \in \mathcal{M}} (\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)) \right\| + \frac{1}{|\mathcal{U}_t|} \left\| \sum_{i \in \mathcal{M} \cap \mathcal{T}_t} (\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)) \right\| \\ &\quad + \frac{1}{|\mathcal{U}_t|} \left\| \sum_{i \in \mathcal{B} \cap \mathcal{T}_t} (\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)) \right\|. \end{aligned}$$

Using Lemma 4.2, we have

$$\frac{1}{|\mathcal{U}_t|} \left\| \sum_{i \in \mathcal{M}} (\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)) \right\| \leq \frac{1-\alpha}{1-\beta} \epsilon_2.$$

with probability exceeding $1 - \frac{2(1-\alpha)md}{(1+n\hat{L}D)^d} - \frac{2d}{(1+(1-\alpha)mn\hat{L}D)^d}$. Also, we obtain

$$\frac{1}{|\mathcal{U}_t|} \left\| \sum_{i \in \mathcal{M} \cap \mathcal{T}_t} (\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)) \right\| \leq \frac{\beta}{1-\alpha} \max_{i \in \mathcal{M}} \|\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)\| \leq \frac{\beta}{1-\alpha} \epsilon_1,$$

with probability at least $1 - \frac{2(1-\alpha)md}{(1+n\hat{L}D)^d}$, where the last inequality is derived from Lemma 4.1. Finally, for the Byzantine term, we have

$$\begin{aligned} \frac{1}{|\mathcal{U}_t|} \left\| \sum_{i \in \mathcal{B} \cap \mathcal{T}_t} (\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)) \right\| &\leq \frac{\alpha}{1-\beta} \max_{i \in \mathcal{B} \cap \mathcal{T}_t} \|\nabla F_i(\mathbf{w}_t)\| + \frac{\alpha}{1-\beta} \|\nabla F(\mathbf{w}_t)\| \\ &\leq \frac{\alpha}{1-\beta} \max_{i \in \mathcal{M}} \|\nabla F_i(\mathbf{w}_t)\| + \frac{\alpha}{1-\beta} \|\nabla F(\mathbf{w}_t)\| \\ &\leq \frac{\alpha}{1-\beta} \max_{i \in \mathcal{M}} \|\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)\| + \frac{2\alpha}{1-\beta} \|\nabla F(\mathbf{w}_t)\| \\ &\leq \frac{\alpha}{1-\beta} \epsilon_1 + \frac{2\alpha}{1-\beta} \|\nabla F(\mathbf{w}_t)\|, \end{aligned}$$

with high probability, where the last inequality follows from Lemma 4.1.

Combining all the terms of T_1 and T_2 , we obtain,

$$\|\Delta\| \leq \frac{\sqrt{1-\delta} + 2\alpha}{1-\beta} \|\nabla F(\mathbf{w}_t)\| + \frac{\sqrt{1-\delta} + \alpha + \beta}{1-\beta} \epsilon_1 + \frac{1-\alpha}{1-\beta} \epsilon_2.$$

Now, using Young's inequality, for any $\lambda > 0$, we obtain

$$\|\Delta\|^2 \leq (1+\lambda) \left(\frac{\sqrt{1-\delta} + 2\alpha}{1-\beta} \right)^2 \|\nabla F(\mathbf{w}_t)\|^2 + \tilde{\epsilon}(\lambda)$$

where

$$\tilde{\epsilon}(\lambda) = 2\left(1 + \frac{1}{\lambda}\right) \left[\left(\frac{\sqrt{1-\delta} + \alpha + \beta}{1-\beta} \right)^2 \epsilon_1^2 + \left(\frac{1-\alpha}{1-\beta} \right)^2 \right] \epsilon_2^2.$$

□

Theorem 4.4. *Suppose Assumptions 4.1, 4.2 and 4.3 hold, and $\alpha \leq \beta < 1/2$. For sufficiently small constant c , we choose the step size $\gamma = \frac{c}{L_F}$. Then, running Algorithm 1 for $T = C_3 \frac{L_F(F(\mathbf{w}_0) - F(\mathbf{w}^*))}{\epsilon}$ iterations yields*

$$\min_{t=0, \dots, T} \|\nabla F(\mathbf{w}_t)\|^2 \leq C \epsilon,$$

with probability greater than or equal to $1 - \frac{c_1(1-\alpha)md}{(1+n\hat{L}D)^d} - \frac{c_2d}{(1+(1-\alpha)mn\hat{L}D)^d}$, provided the compression factor satisfies $\delta > \delta_0 + 4\alpha - 9\alpha^2 + 4\alpha^3$, where $\delta_0 = \left(1 - \frac{(1-\beta)^2}{1+\lambda_0}\right)$ and λ_0 is a (sufficiently small) positive constant.

Proof. Proof of Theorem 4.4

Let $g(\mathbf{w}_t) = \frac{1}{|\mathcal{U}_t|} \sum_{i \in \mathcal{U}_t} \mathcal{Q}(\nabla F_i(\mathbf{w}_t))$ and $\Delta = g(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)$. We Lemma 4.3 to control of $\|\Delta\|^2$.

We first show that with Assumption 4.3 and with the choice of step size γ , we always stay in \mathcal{W} without projection. Recall that $g(\mathbf{w}_t) = \frac{1}{|\mathcal{U}_t|} \sum_{i \in \mathcal{U}_t} \mathcal{Q}(\nabla F_i(\mathbf{w}_t))$ and $\Delta = g(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)$. We have

$$\begin{aligned} \|\mathbf{w}_{t+1} - \mathbf{w}^*\| &\leq \|\mathbf{w}_t - \mathbf{w}^*\| + \gamma(\|\nabla F(\mathbf{w}_t)\| + \|g(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)\|) \\ &\leq \|\mathbf{w}_t - \mathbf{w}^*\| + \frac{c}{L_F}(\|\nabla F(\mathbf{w}_t)\| + \|\Delta\|) \end{aligned}$$

We use Lemma 4.3 with $\lambda = \lambda_0$ for a sufficiently small positive constant λ_0 . Define $\delta_0 = \left(1 - \frac{(1-\beta)^2}{1+\lambda_0}\right)$. A little algebra shows that provided $\delta > \delta_0 + 4\alpha - 9\alpha^2 + 4\alpha^3$, we obtain

$$\|\Delta\|^2 \leq (1 - c_0)\|\nabla F(\mathbf{w}_t)\|^2 + \epsilon$$

with probability greater than or equal to $1 - \frac{c_1(1-\alpha)md}{(1+n\hat{L}D)^d} - \frac{c_2d}{(1+(1-\alpha)mn\hat{L}D)^d}$, where c_0 is a positive constant and ϵ is defined in equation (4.4). Substituting, we obtain

$$\|\mathbf{w}_{t+1} - \mathbf{w}^*\| \leq \|\mathbf{w}_t - \mathbf{w}^*\| + \frac{c_1}{L_F} \left((1 + \sqrt{1 - c_0})\|\nabla F(\mathbf{w}_t)\| + \sqrt{\epsilon} \right)$$

$$\leq \|\mathbf{w}_t - \mathbf{w}^*\| + \frac{c_1}{L_F} \left((2 - \frac{c_0}{2}) \|\nabla F(\mathbf{w}_t)\| + \sqrt{\epsilon} \right).$$

where we use the fact that $\sqrt{1 - c_0} \leq 1 - c_0/2$. Now, running $T = C \frac{L_F(F(\mathbf{w}_0) - F(\mathbf{w}^*))}{\epsilon}$ iterations, we see that Assumption 4.3 ensures that the iterations of Algorithm 1 is always in \mathcal{W} . Hence, let us now analyze the algorithm without the projection step.

Using the smoothness of $F(\cdot)$, we have

$$F(\mathbf{w}_{t+1}) \leq F(\mathbf{w}_t) + \langle \nabla F(\mathbf{w}_t), \mathbf{w}_{t+1} - \mathbf{w}_t \rangle + \frac{L_F}{2} \|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2.$$

Using the iteration of Algorithm 1, we obtain

$$\begin{aligned} F(\mathbf{w}_{t+1}) &\leq F(\mathbf{w}_t) - \gamma \langle \nabla F(\mathbf{w}_t), \nabla F(\mathbf{w}_t) + \Delta \rangle + \frac{\gamma^2 L_F}{2} \|\nabla F(\mathbf{w}_t) + \Delta\|^2 \\ &\leq F(\mathbf{w}_t) - \gamma \|\nabla F(\mathbf{w}_t)\|^2 - \gamma \langle \nabla F(\mathbf{w}_t), \Delta \rangle \\ &\quad + \frac{\gamma^2 L_F}{2} \|\nabla F(\mathbf{w}_t)\|^2 + \frac{\gamma^2 L_F}{2} \|\Delta\|^2 + \gamma^2 L_F \langle \nabla F(\mathbf{w}_t), \Delta \rangle \\ &\leq F(\mathbf{w}_t) - (\gamma - \frac{\gamma^2 L_F}{2}) \|\nabla F(\mathbf{w}_t)\|^2 \\ &\quad + (\gamma + \gamma^2 L_F) \left(\frac{\rho}{2} \|\nabla F(\mathbf{w}_t)\|^2 + \frac{1}{2\rho} \|\Delta\|^2 \right) + \frac{\gamma^2 L_F}{2} \|\Delta\|^2, \end{aligned}$$

where $\rho > 0$ and the last inequality follows from Young's inequality. Substituting $\rho = 1$, we obtain

$$(\gamma/2 - \gamma^2 L_F) \|\nabla F(\mathbf{w}_t)\|^2 \leq F(\mathbf{w}_t) - F(\mathbf{w}_{t+1}) + (\gamma/2 + \gamma^2 L_F) \|\Delta\|^2.$$

We now use Lemma 4.3 to obtain

$$\begin{aligned} (\frac{\gamma}{2} - \gamma^2 L_F) \|\nabla F(\mathbf{w}_t)\|^2 &\leq F(\mathbf{w}_t) - F(\mathbf{w}_{t+1}) \\ &\quad + (\gamma/2 + \gamma^2 L_F) \left((1 + \lambda) \left(\frac{\sqrt{1 - \delta} + 2\alpha}{1 - \beta} \right)^2 \|\nabla F(\mathbf{w}_t)\|^2 + \tilde{\epsilon}(\lambda) \right). \end{aligned}$$

with high probability. Upon further simplification, we have

$$\begin{aligned} \left(\frac{\gamma}{2} - \frac{\gamma}{2}(1+\lambda) \left(\frac{\sqrt{1-\delta} + 2\alpha}{1-\beta} \right)^2 - (1+\lambda) \left(\frac{\sqrt{1-\delta} + 2\alpha}{1-\beta} \right)^2 \gamma^2 L_F - \gamma^2 L_F \right) \|\nabla F(\mathbf{w}_t)\|^2 \\ \leq F(\mathbf{w}_t) - F(\mathbf{w}_{t+1}) + (\gamma/2 + \gamma^2 L_F) \tilde{\epsilon}(\lambda). \end{aligned}$$

We now substitute $\gamma = \frac{c}{L_F}$, for a small enough constant c , so that we can ignore the contributions of the terms with quadratic dependence on γ . We substitute $\lambda = \lambda_0$ for a sufficiently small positive constant λ_0 . Provided $\delta > \delta_0 + 4\alpha - 9\alpha^2 + 4\alpha^3$, where $\delta_0 = \left(1 - \frac{(1-\beta)^2}{1+\lambda_0}\right)^2$, we have

$$\left(\frac{\gamma}{2} - \frac{\gamma}{2}(1+\lambda) \left(\frac{\sqrt{1-\delta} + 2\alpha}{1-\beta} \right)^2 - (1+\lambda) \left(\frac{\sqrt{1-\delta} + 2\alpha}{1-\beta} \right)^2 \gamma^2 L_F - \gamma^2 L_F \right) = \frac{c_1}{L_F},$$

where c_1 is a constant. With this choice, we obtain

$$\frac{1}{T+1} \sum_{t=0}^T \|\nabla F(\mathbf{w}_t)\|^2 \leq C_1 \frac{L_F(F(\mathbf{w}_0) - F(\mathbf{w}^*))}{T+1} + C_2 \epsilon$$

where the first term is obtained from a telescopic sum and ϵ is defined in equation (4.4).

Finally, we obtain

$$\min_{t=0, \dots, T} \|\nabla F(\mathbf{w}_t)\|^2 \leq C_1 \frac{L_F(F(\mathbf{w}_0) - F(\mathbf{w}^*))}{T+1} + C_2 \epsilon$$

with probability greater than or equal to $1 - \frac{c_1(1-\alpha)md}{(1+n\hat{L}D)^d} - \frac{c_2d}{(1+(1-\alpha)mn\hat{L}D)^d}$, proving Theorem 4.4. \square

A few remarks are in order. In the following remarks, we fix the dimension d , and discuss the dependence of ϵ on (α, δ, n, m) .

Remark 4.1. (Rate of Convergence) Algorithm 1 with T iterations yields

$$\min_{t=0,\dots,T} \|\nabla F(\mathbf{w}_t)\|^2 \leq \frac{C_1 L_F (F(\mathbf{w}_0) - F(\mathbf{w}^*))}{T+1} + C_2 \epsilon$$

with high probability. We see that Algorithm 1 converges at a rate of $\mathcal{O}(1/T)$, and finally plateaus at an error floor of ϵ . Note that the rate of convergence is same as [135]. Hence, even with compression, the (order-wise) convergence rate is unaffected.

Remark 4.2. We observe, from the definition of ϵ that the price for compression is $\tilde{\mathcal{O}}(\frac{1-\delta}{n})$.

Remark 4.3. Substituting $\delta = 1$ (no compression) in ϵ , we get $\epsilon = \tilde{\mathcal{O}}(\frac{\alpha^2}{n} + \frac{1}{mn})$, which matches the (statistical) rate of [135]. A simple norm based thresholding operation is computationally simple and efficient in the high dimensional settings compared to the coordinate wise median and trimmed mean to achieve robustness and obtain the the same statistical error and iteration complexity as [135]

Remark 4.4. When the compression factor δ is large enough, satisfying $\delta \geq 1 - \alpha^2$, we obtain $\epsilon = \tilde{\mathcal{O}}(\frac{\alpha^2}{n} + \frac{1}{mn})$. In this regime, the iteration complexity and the final statistical error of Algorithm 1 is order-wise identical to the setting with no compression [135]. We emphasize here that a reasonable high δ is often observed in practical applications like training of neural nets [71, Figure 2].

Remark 4.5. (Optimality) For a distributed mean estimation problem, Observation 1 in [135] implies that any algorithm will yield an (statistical) error of $\Omega(\frac{\alpha^2}{n} + \frac{d}{mn})$. Hence, in the regime where $\delta \geq 1 - \alpha^2$, our error-rate is optimal.

Remark 4.6. For the convergence of Algorithm 1, we require $\delta > \delta_0 + 4\alpha - 9\alpha^2 + 4\alpha^3$, implying that our analysis will not work if δ is very close to 0. Note that a very small δ does not give good accuracy in practical applications [71, Figure 2]. Also, note that, from the definition of δ_0 , we can choose λ_0 sufficiently small at the expense of

increasing the multiplicative constant in ϵ by a factor of $1/\lambda_0$. Since the error-rate considers asymptotic in m and n , increasing a constant factor is insignificant. A sufficiently small λ_0 implies $\delta_0 = \mathcal{O}(2\beta)$, and hence we require $\delta > 4\alpha + 2\beta$ (ignoring the higher order dependence).

Remark 4.7. The requirement $\delta > 4\alpha + 2\beta$ can be seen as a trade-off between the amount of compression and the fraction of adversaries in the system. As α increases, the amount of (tolerable) compression decreases and vice versa.

4.5 Distributed Optimization with Arbitrary Adversaries

In this section we remove the assumption of restricted adversary (as in Section 4.4) and make the learning algorithm robust to the adversarial effects of both the computation and compression unit. In particular, here we consider Algorithm 1 with Option II. Hence, the Byzantine machines do not need to adhere to the mandated compression algorithm.

In Option II, the worker machines send $\mathcal{Q}(\nabla F_i(\mathbf{w}_t))$ to the center machine. The center machine computes its norm, and discards the top β fraction of the worker machines having largest norm. Note that it is crucial that the center machine computes the norm of $\mathcal{Q}(\nabla F_i(\mathbf{w}_t))$, instead of asking the worker machine to send it (similar to Option I). Otherwise, a Byzantine machine having a large $\|\mathcal{Q}(\mathbf{x})\|_q$ can (wrongly) report a small value of $\|\mathcal{Q}(\mathbf{x})\|_q$, gets selected in the trimming phase and influences (or can potentially diverge) the optimization algorithm. Hence, the center needs to compute $\|\mathcal{Q}(\mathbf{x})\|_q$ to remove such issues.

Although this framework is more general in terms of Byzantine attacks, however, in this setting, the statistical error-rate of our proposed algorithm is slightly weaker than that of Theorem 4.4. Furthermore, the (δ, α) trade-off is stricter compared to Theorem 4.4.

4.5.1 Main Results

We continue to assume that the population loss function $F(\cdot)$ is smooth and non-convex and analyze Algorithm 1 with Option II. We have the following result. For the clarity of exposition, we define the following quantity which will be used in the results of this section:

$$\tilde{\epsilon} = 2(1 + \frac{1}{\lambda_0}) \left(\left(\frac{(1 + \beta)\sqrt{1 - \delta} + \alpha + \beta}{1 - \beta} \right)^2 \epsilon_1^2 + \left(\frac{1 - \alpha}{1 - \beta} \right)^2 \epsilon_2^2 \right).$$

Comparing $\tilde{\epsilon}$ with ϵ , we observe that $\tilde{\epsilon} > \epsilon$. Also, note that,

$$\tilde{\epsilon} = \tilde{\mathcal{O}} \left(d^2 \left[\frac{\alpha^2}{n} + \frac{1 - \delta}{n} + \frac{1}{mn} \right] \right), \quad (4.7)$$

which suggests that $\tilde{\epsilon}$ and ϵ are order-wise similar. We have the following assumption, which parallels Assumption 4.3, with ϵ replaced by $\tilde{\epsilon}$.

Assumption 4.4. (*Size of parameter space \mathcal{W}*) Suppose that $\|\nabla F(\mathbf{w})\| \leq M$ for all $\mathbf{w} \in \mathcal{W}$. We assume that \mathcal{W} contains the ℓ_2 ball $\{\mathbf{w} : \|\mathbf{w} - \mathbf{w}_0\| \leq c[(2 - \frac{c_0}{2})M + \sqrt{\tilde{\epsilon}}] \frac{F(\mathbf{w}_0) - F(\mathbf{w}^*)}{\tilde{\epsilon}}\}$, where c_0 is a constant, δ is the compression factor and $\tilde{\epsilon}$ is defined in equation (4.7).

Lemma 4.5. For any $\lambda > 0$, we have,

$$\|\tilde{\Delta}\|^2 \leq ((1 + \lambda) \left(\frac{(1 + \beta)\sqrt{1 - \delta} + 2\alpha}{1 - \beta} \right)^2 \|\nabla F(\mathbf{w}_t)\|^2 + \tilde{\epsilon}(\lambda))$$

with probability greater than or equal to $1 - \frac{c_1(1 - \alpha)md}{(1 + n\hat{L}D)^d} - \frac{c_2d}{(1 + (1 - \alpha)mn\hat{L}D)^d}$, where

$$\tilde{\epsilon}(\lambda) = 2(1 + \frac{1}{\lambda}) \left(\left(\frac{(1 + \beta)\sqrt{1 - \delta} + \alpha + \beta}{1 - \beta} \right)^2 \epsilon_1^2 + \left(\frac{1 - \alpha}{1 - \beta} \right)^2 \epsilon_2^2 \right).$$

with ϵ_1 and ϵ_2 as defined in equation (4.2) and (4.3) respectively.

Proof. Proof of Lemma 4.5

Here we prove an upper bound on the norm of

$$\tilde{\Delta} = g(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)$$

where $g(\mathbf{w}_t) = \frac{1}{|\mathcal{U}_t|} \sum_{i \in \mathcal{U}_t} Q(\nabla F_i(\mathbf{w}_t))$.

We have

$$\begin{aligned} \|\tilde{\Delta}\| &= \left\| \frac{1}{|\mathcal{U}_t|} \sum_{i \in \mathcal{U}_t} Q(\nabla F_i(\mathbf{w}_t)) - \nabla F(\mathbf{w}_t) \right\| \\ &= \frac{1}{|\mathcal{U}_t|} \left\| \sum_{i \in \mathcal{M}} [Q(\nabla F_i(\mathbf{w}_t)) - \nabla F(\mathbf{w}_t)] - \sum_{i \in (\mathcal{M} \cap \mathcal{T}_t)} [Q(\nabla F_i(\mathbf{w}_t)) - \nabla F(\mathbf{w}_t)] \right. \\ &\quad \left. + \sum_{i \in (\mathcal{B} \cap \mathcal{U}_t)} [Q(\nabla F_i(\mathbf{w}_t)) - \nabla F(\mathbf{w}_t)] \right\| \\ &\leq \frac{1}{|\mathcal{U}_t|} \left(\underbrace{\left\| \sum_{i \in \mathcal{M}} Q(\nabla F_i(\mathbf{w}_t)) - \nabla F(\mathbf{w}_t) \right\|}_{T_1} + \underbrace{\left\| \sum_{i \in (\mathcal{M} \cap \mathcal{T}_t)} Q(\nabla F_i(\mathbf{w}_t)) - \nabla F(\mathbf{w}_t) \right\|}_{T_2} \right. \\ &\quad \left. + \underbrace{\left\| \sum_{i \in (\mathcal{B} \cap \mathcal{U}_t)} Q(\nabla F_i(\mathbf{w}_t)) - \nabla F(\mathbf{w}_t) \right\|}_{T_3} \right) \end{aligned}$$

Now we bound each term separately. For the first term, we have

$$\begin{aligned} \frac{1}{|\mathcal{U}_t|} T_1 &= \frac{1}{|\mathcal{U}_t|} \left\| \sum_{i \in \mathcal{M}} Q(\nabla F_i(\mathbf{w}_t)) - \nabla F(\mathbf{w}_t) \right\| \\ &= \frac{1}{|\mathcal{U}_t|} \left\| \sum_{i \in \mathcal{M}} Q(\nabla F_i(\mathbf{w}_t)) - \nabla F_i(\mathbf{w}_t) \right\| + \frac{1}{|\mathcal{U}_t|} \left\| \sum_{i \in \mathcal{M}} \nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t) \right\| \\ &\leq \frac{1}{|\mathcal{U}_t|} \sum_{i \in \mathcal{M}} \left(\|Q(\nabla F_i(\mathbf{w}_t)) - \nabla F_i(\mathbf{w}_t)\| \right) + \frac{1 - \alpha}{1 - \beta} \epsilon_2 \\ &\leq \frac{1}{|\mathcal{U}_t|} \sum_{i \in \mathcal{M}} \left(\sqrt{1 - \delta} \|\nabla F_i(\mathbf{w}_t)\| \right) + \frac{1 - \alpha}{1 - \beta} \epsilon_2 \\ &\leq \frac{\sqrt{1 - \delta}}{|\mathcal{U}_t|} \sum_{i \in \mathcal{M}} \left(\|\nabla F(\mathbf{w}_t)\| + \|\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)\| \right) + \frac{1 - \alpha}{1 - \beta} \epsilon_2 \end{aligned}$$

$$\leq \frac{\sqrt{1-\delta}(1-\alpha)}{1-\beta} \|\nabla F(\mathbf{w}_t)\| + \frac{\sqrt{1-\delta}(1-\alpha)}{1-\beta} \epsilon_1 + \frac{1-\alpha}{1-\beta} \epsilon_2$$

where we use the definition of a δ -approximate compressor, Lemma 4.1 and Lemma 4.2.

Similarly, we can bound T_2 as

$$\begin{aligned} T_2 &\leq \sum_{i \in (\mathcal{M} \cap \mathcal{T}_t)} \|Q(\nabla F_i(\mathbf{w}_t)) - \nabla F(\mathbf{w}_t)\| \\ &\leq \beta m \max_{i \in \mathcal{M}} \|Q(\nabla F_i(\mathbf{w}_t)) - \nabla F(\mathbf{w}_t)\| \\ &\leq \beta m \max_{i \in \mathcal{M}} \left(\sqrt{1-\delta} \|\nabla F_i(\mathbf{w}_t)\| + \|\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)\| \right) \\ &\leq \beta m \max_{i \in \mathcal{M}} \left(\sqrt{1-\delta} \|\nabla F(\mathbf{w}_t)\| + (1 + \sqrt{1-\delta}) \|\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)\| \right) \end{aligned}$$

where we use the definition of δ -approximate compressor. Hence invoking Lemma 4.1, we obtain

$$\frac{1}{|\mathcal{U}_t|} T_2 \leq \frac{\beta \sqrt{1-\delta}}{1-\beta} \|\nabla F(\mathbf{w}_t)\| + \frac{\beta(1 + \sqrt{1-\delta})}{1-\beta} \epsilon_1$$

Also, owing to the trimming with $\beta > \alpha$, we have at least one good machine in the set \mathcal{T}_t for all t . Now each term in the set $\mathcal{B} \cap \mathcal{U}_t$, we have

$$\begin{aligned} T_3 &= \sum_{i \in (\mathcal{B} \cap \mathcal{U}_t)} \|Q(\nabla F_i(\mathbf{w}_t)) - \nabla F(\mathbf{w}_t)\| \\ &\leq \alpha m (\max_{i \in \mathcal{M}} \|Q(\nabla F_i(\mathbf{w}_t))\| + \|\nabla F(\mathbf{w}_t)\|) \\ &\leq \alpha m (\max_{i \in \mathcal{M}} \sqrt{1-\delta} \|\nabla F_i(\mathbf{w}_t)\| + \|\nabla F_i(\mathbf{w}_t)\| + \|\nabla F(\mathbf{w}_t)\|) \\ &\leq \alpha m \left((1 + \sqrt{1-\delta}) \epsilon_1 + (2 + \sqrt{1-\delta}) \|\nabla F(\mathbf{w}_t)\| \right) \\ \frac{1}{|\mathcal{U}_t|} T_3 &\leq \frac{\alpha(2 + \sqrt{1-\delta})}{1-\beta} \|\nabla F(\mathbf{w}_t)\| + \frac{\alpha(1 + \sqrt{1-\delta})}{1-\beta} \epsilon_1 \end{aligned}$$

where we use Lemma 4.1. Putting T_1, T_2, T_3 we get

$$\begin{aligned}
\|\tilde{\Delta}\| &\leq \left(\frac{\sqrt{1-\delta}(1-\alpha)}{1-\beta} + \frac{\beta\sqrt{1-\delta}}{1-\beta} + \frac{\alpha(2+\sqrt{1-\delta})}{1-\beta} \right) \|\nabla F(\mathbf{w}_t)\| \\
&\quad + \left(\frac{\sqrt{1-\delta}(1-\alpha)}{1-\beta} + \frac{\beta(1+\sqrt{1-\delta})}{1-\beta} + \frac{\alpha(1+\sqrt{1-\delta})}{1-\beta} \right) \epsilon_1 + \frac{1-\alpha}{1-\beta} \epsilon_2 \\
&= \left(\frac{(1+\beta)\sqrt{1-\delta}+2\alpha}{1-\beta} \right) \|\nabla F(\mathbf{w}_t)\| + \left(\frac{(1+\beta)\sqrt{1-\delta}+\alpha+\beta}{1-\beta} \right) \epsilon_1 + \frac{1-\alpha}{1-\beta} \epsilon_2 \\
\|\tilde{\Delta}\|^2 &\leq (1+\lambda) \left(\frac{(1+\beta)\sqrt{1-\delta}+2\alpha}{1-\beta} \right)^2 \|\nabla F(\mathbf{w}_t)\|^2 + \tilde{\epsilon}(\lambda)
\end{aligned}$$

where $\tilde{\epsilon}(\lambda) = 2(1 + \frac{1}{\lambda}) \left(\left(\frac{(1+\beta)\sqrt{1-\delta}+\alpha+\beta}{1-\beta} \right)^2 \epsilon_1^2 + \left(\frac{1-\alpha}{1-\beta} \right)^2 \epsilon_2^2 \right)$. Hence, the lemma follows. \square

Theorem 4.6. *Suppose Assumptions 4.1, 4.2 and 4.4 hold, and $\alpha \leq \beta < 1/2$. For sufficiently small constant c , we choose the step size $\gamma = \frac{c}{L_F}$. Then, running Algorithm 1 for $T = C_3 \frac{L_F(F(\mathbf{w}_0) - F(\mathbf{w}^*))}{\tilde{\epsilon}}$ iterations yields*

$$\min_{t=0, \dots, T} \|\nabla F(\mathbf{w}_t)\|^2 \leq C \tilde{\epsilon},$$

with probability greater than or equal to $1 - \frac{c_1(1-\alpha)md}{(1+n\hat{L}D)^d} - \frac{c_2d}{(1+(1-\alpha)mn\hat{L}D)^d}$, provided the compression factor satisfies $\delta > \tilde{\delta}_0 + 4\alpha - 8\alpha^2 + 4\alpha^3$, where $\tilde{\delta}_0 = \left(1 - \frac{(1-\beta)^2}{(1+\beta)^2(1+\lambda_0)} \right)$ and λ_0 is a (sufficiently small) positive constant.

Proof. Proof of Theorem 4.6 The proof of convergence for Theorem 4.6 follows the same steps as Theorem 4.4. Recall that the quantity of interest is

$$\tilde{\Delta} = g(\mathbf{w}_t) - \nabla F(\mathbf{w}_t).$$

The proof parallels the proof of 4.4, except the fact that we use Lemma 4.5 to upper bound $\|\tilde{\Delta}\|^2$. Correspondingly, a little algebra shows that we require

$\delta > \tilde{\delta}_0 + 4\alpha - 8\alpha^2 + 4\alpha^3$, where $\tilde{\delta}_0 = \left(1 - \frac{(1-\beta)^2}{(1+\beta)^2(1+\lambda_0)}\right)$, where λ_0 is a sufficiently small positive constant. With the above requirement, the proof follows the same steps as Theorem 4.4 and hence we omit the details here.

□

Remark 4.8. *The above result and their consequences resemble that of Theorem 4.4. Since $\tilde{\epsilon} > \epsilon$, the statistical error-rate in Theorem 4.6 is strictly worse than that of Theorem 4.4 (although order-wise they are same).*

Remark 4.9. *Note that the definition of δ_0 is different than in Theorem 4.4. For a sufficiently small λ_0 , we see $\tilde{\delta}_0 = \mathcal{O}(4\beta)$, which implies we require $\delta > 4\beta + 4\alpha$ for the convergence of Theorem 4.6. Note that this is a slightly strict requirement compared to Theorem 4.4. In particular, for a given δ , Algorithm 1 with Option II can tolerate less number of Byzantine machines compared to Option I.*

Remark 4.10. *The result in Theorem 4.6 is applicable for arbitrary adversaries, whereas Theorem 4.4 relies on the adversary being restrictive. Hence, we can view the limitation of Theorem 4.6 (such as worse statistical error-rate and stricter (δ, α) trade-off) as a price of accommodating arbitrary adversaries.*

4.6 Byzantine Robust Distributed Learning with Error Feedback

We now investigate the role of error feedback [71] in distributed learning with Byzantine worker machines.

In order to address the issues of convergence for sign based algorithms (like *signSGD*), [71] proposes a class of optimization algorithms that uses *error feedback*. In this setting, the worker machine locally stores the error between the actual local gradient and its compressed counterpart. Using this as feedback, the worker machine adds this error term to the compressed gradient in the subsequent iteration. Intuitively,

Algorithm 2 Distributed Compressed Gradient Descent with Error Feedback

- 1: **Input:** Step size γ , Compressor $\mathcal{Q}(\cdot)$, parameter $\beta(> \alpha)$.
 - 2: **Initialize:** Initial iterate \mathbf{w}_0 , $\mathbf{e}_i(0) = 0 \ \forall i \in [m]$
 - 3: **for** $t = 0, 1, \dots, T - 1$ **do**
 - 4: Central machine: sends \mathbf{w}_t to all worker
 for $i \in [m]$ **do in parallel**
 - 5: i -th non-Byzantine worker machine:
 - computes $\mathbf{p}_i(\mathbf{w}_t) = \gamma \nabla F_i(\mathbf{w}_t) + \mathbf{e}_i(t)$
 - sends $\mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t))$ to the central machine
 - computes $\mathbf{e}_i(t + 1) = \mathbf{p}_i(\mathbf{w}_t) - \mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t))$
 - 6: Byzantine worker machine:
 - sends \star to the central machine.
 - 7: At Central machine:
 - sorts the worker machines in non-decreasing order according to $\|\mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t))\|$.
 - returns the indices of the first $1 - \beta$ fraction of elements as \mathcal{U}_t .
 - $\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\gamma}{|\mathcal{U}_t|} \sum_{i \in \mathcal{U}_t} \mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t))$
 - 8: **end for**
-

this accounts for correcting the the direction of the local gradient. The error-feedback has its roots in some of the classical communication system like “delta-sigma” modulator and adaptive modulator([60]).

We analyze the distributed error feedback algorithm in the presence of Byzantine machines. The algorithm is presented in Algorithm 2. We observe that here the central machine sorts the worker machines according to the norm of the compressed local gradients, and ignore the largest β fraction.

Note that, similar to Section 4.5, we handle arbitrary adversaries. In the subsequent section, we show (both theoretically and experimentally) that the statistical error rate of Algorithm 2 is smaller than Algorithm 1.

4.6.1 Main Results

In this section we analyze Algorithm 2 and obtain the rate of the convergence under non-convex smooth loss functions. Throughout the section, we select γ as the step size and assume that Algorithm 2 is run for T iterations. We start with the following assumption.

Assumption 4.5. *For all non-Byzantine worker machine i , the local loss functions $F_i(\cdot)$ satisfy $\|\nabla F_i(\mathbf{x})\|^2 \leq \sigma^2$, where $x \in \{\mathbf{w}_j\}_{j=0}^T$, and $\{\mathbf{w}_0, \dots, \mathbf{w}_T\}$ are the iterates of Algorithm 2.*

Note that since $F_i(\cdot)$ can be written as loss over data points of machine i , we observe that the bounded gradient condition is equivalent to the bounded second moment condition for SGD, and have featured in several previous works, see, e.g., [69], [91]. Here, we are using all the data points and (hence no randomness over the choice of data points) perform gradient descent instead of SGD. Also, note that Assumption 4.5 is weaker than the bounded second moment condition since we do not require $\|\nabla F_i(\mathbf{x})\|^2$ to be bounded for all \mathbf{x} ; just when $\mathbf{x} \in \{\mathbf{w}_j\}_{j=0}^T$.

We also require the following assumption on the size of the parameter space \mathcal{W} , which parallels Assumption 4.3 and 4.4.

Assumption 4.6. *(Size of parameter space \mathcal{W}) Suppose that $\|\nabla F(\mathbf{w})\| \leq M$ for all $\mathbf{w} \in \mathcal{W}$. We assume that \mathcal{W} contains the ℓ_2 ball $\{\mathbf{w} : \|\mathbf{w} - \mathbf{w}_0\| \leq \gamma r^* T\}$, where*

$$r^* = \epsilon_2 + M + \frac{6\beta(1 + \sqrt{1 - \delta})}{(1 - \beta)} \left(\epsilon_1 + M + \sqrt{\frac{3(1 - \delta)}{}} \delta \sigma \right) + \sqrt{\frac{12(1 - \delta)}{}} \delta \sigma,$$

and (ϵ_1, ϵ_2) are defined in equations (4.2) and (4.3) respectively.

Similar to Assumption 4.3 and 4.4, we use the above assumption to ensure that the iterates of Algorithm 2 stays in \mathcal{W} , and we emphasize that this is a standard assumption to control the iterates for non-convex loss function (see [135, 136]).

To simplify notation and for the clarity of exposition, we define the following quantities which will be used in the main results of this section.

$$\Delta_1 = \frac{9(1 + \sqrt{1 - \delta})^2}{2c(1 - \beta)^2} [\alpha^2 + \beta^2 + (\beta - \alpha)^2] \left(\epsilon_1^2 + \frac{3(1 - \delta)}{\delta} \sigma^2 \right) + \frac{50}{c} \epsilon_2^2, \quad (4.8)$$

$$\begin{aligned} \Delta_2 = & \frac{L^2}{2} \frac{3(1 - \delta)\sigma^2}{c\delta} + \frac{2L\epsilon_2^2}{c} \\ & + \left(\frac{1}{2} + L \right) \frac{9(1 + \sqrt{1 - \delta})^2}{c(1 - \beta)^2} [\alpha^2 + \beta^2 + (\beta - \alpha)^2] \left(\epsilon_1^2 + \frac{3(1 - \delta)}{\delta} \sigma^2 \right), \end{aligned} \quad (4.9)$$

$$\Delta_3 = \left(\frac{L^2}{100} + 25L^2 \right) \frac{3(1 - \delta)\sigma^2}{c\delta}, \quad (4.10)$$

where c is a universal constant.

We show the following rate of convergence to a critical point of the population loss function $F(\cdot)$.

Theorem 4.7. *Suppose Assumptions 4.1, 4.2, 4.5 and 4.6 hold, and $\alpha \leq \beta < 1/2$. Then, running Algorithm 1 for T iterations with step size γ yields*

$$\min_{t=0, \dots, T} \|\nabla F(\mathbf{w}_t)\|^2 \leq \frac{F(\mathbf{w}_0) - F^*}{c\gamma(T + 1)} + \Delta_1 + \gamma\Delta_2 + \gamma^2\Delta_3,$$

with probability greater than or equal to $1 - \frac{c_1(1-\alpha)md}{(1+n\hat{L}D)^d} - \frac{c_2d}{(1+(1-\alpha)mn\hat{L}D)^d}$, provided the compression factor satisfies $\frac{(1+\sqrt{1-\delta})^2}{(1-\beta)^2} [\alpha^2 + \beta^2 + (\beta - \alpha)^2] < 0.107$. Here Δ_1, Δ_2 and Δ_3 are defined in equations (4.8), (4.9) and (4.10) respectively.

Proof. Proof of Theorem 4.7

We first define an auxiliary sequence defined as:

$$\tilde{\mathbf{w}}_t = \mathbf{w}_t - \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathbf{e}_i(t)$$

Hence, we obtain

$$\tilde{\mathbf{w}}_{t+1} = \mathbf{w}_{t+1} - \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathbf{e}_i(t+1).$$

For notational simplicity, let us drop the subscript t from \mathcal{U}_t and \mathcal{T}_t and denote them as \mathcal{U} and \mathcal{T} .

Since (we will ensure that the iterates remain in the parameter space and hence we can ignore the projection step),

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} \mathbf{p}_i(\mathbf{w}_t),$$

we get

$$\begin{aligned} \tilde{\mathbf{w}}_{t+1} &= \mathbf{w}_t - \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} \mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t)) - \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathbf{e}_i(t+1) \\ &= \mathbf{w}_t - \frac{1}{|\mathcal{U}|} \left(\sum_{i \in \mathcal{M}} \mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t)) + \sum_{i \in \mathcal{B} \cap \mathcal{U}} \mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t)) - \sum_{i \in \mathcal{M} \cap \mathcal{T}} \mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t)) \right) - \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathbf{e}_i(t+1) \\ &= \mathbf{w}_t - \left(\frac{1-\alpha}{1-\beta} \right) \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t)) - \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathbf{e}_i(t+1) - \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{B} \cap \mathcal{U}} \mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t)) \\ &\quad + \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{M} \cap \mathcal{T}} \mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t)) \end{aligned}$$

Since $\mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t)) + \mathbf{e}_i(t+1) = \mathbf{p}_i(\mathbf{w}_t)$ for all $i \in \mathcal{M}$, we obtain

$$\left(\frac{1-\alpha}{1-\beta} \right) \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t)) + \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathbf{e}_i(t+1) = \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathbf{p}_i(\mathbf{w}_t) + \frac{\beta-\alpha}{1-\beta} \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t))$$

Let us denote $T_1 = \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{B} \cap \mathcal{U}} \mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t))$, $T_2 = \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{M} \cap \mathcal{T}} \mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t))$ and $T_3 = \frac{\beta-\alpha}{1-\beta} \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t))$. With this, we obtain

$$\begin{aligned} \tilde{\mathbf{w}}_{t+1} &= \mathbf{w}_t - \frac{1}{|\mathcal{M}|} \mathbf{p}_i(\mathbf{w}_t) - T_1 + T_2 - T_3 \\ &= \tilde{\mathbf{w}}_t + \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathbf{e}_i(t) - \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathbf{p}_i(\mathbf{w}_t) - \tilde{T} \\ &= \tilde{\mathbf{w}}_t - \gamma \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \nabla F_i(\mathbf{w}_t) - \tilde{T} \end{aligned}$$

where $\tilde{T} = T_1 - T_2 + T_3$. Observe that the auxiliary sequence looks similar to a distributed gradient step with a presence of \tilde{T} . For the convergence analysis, we will use this relation along with an upper bound on $\|\tilde{T}\|$.

Using this auxiliary sequence, we first ensure that the iterates of our algorithm remains close to one another. To that end, we have

$$\begin{aligned} \mathbf{w}_{t+1} - \mathbf{w}_t &= \tilde{\mathbf{w}}_{t+1} - \tilde{\mathbf{w}}_t + \frac{1}{|\mathcal{M}|} \mathbf{e}_i(t+1) - \frac{1}{|\mathcal{M}|} \mathbf{e}_i(t) \\ &= -\gamma \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \nabla F_i(\mathbf{w}_t) - \tilde{T} + \frac{1}{|\mathcal{M}|} \mathbf{e}_i(t+1) - \frac{1}{|\mathcal{M}|} \mathbf{e}_i(t). \end{aligned}$$

Hence, we obtain

$$\begin{aligned} \|\mathbf{w}_{t+1} - \mathbf{w}_t\| &\leq \left\| \gamma \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \nabla F_i(\mathbf{w}_t) \right\| + \|\tilde{T}\| + \left\| \frac{1}{|\mathcal{M}|} \mathbf{e}_i(t+1) \right\| + \left\| \frac{1}{|\mathcal{M}|} \mathbf{e}_i(t) \right\| \\ &\leq \gamma \left\| \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t) \right\| + \gamma \|\nabla F(\mathbf{w}_t)\| + \|\tilde{T}\| + \left\| \frac{1}{|\mathcal{M}|} \mathbf{e}_i(t+1) \right\| + \left\| \frac{1}{|\mathcal{M}|} \mathbf{e}_i(t) \right\| \\ &\leq \gamma \epsilon_2 + \gamma \|\nabla F(\mathbf{w}_t)\| + \|\tilde{T}\| + \left\| \frac{1}{|\mathcal{M}|} \mathbf{e}_i(t+1) \right\| + \left\| \frac{1}{|\mathcal{M}|} \mathbf{e}_i(t) \right\|. \end{aligned}$$

Now, using Lemma 4.8 and Lemma 4.9 in conjunction with Assumption 4.3 ensures the iterates of Algorithm 2 stays in the parameter space \mathcal{W} .

We assume that the global loss function $F(\cdot)$ is L_F smooth. We get

$$F(\tilde{\mathbf{w}}_{t+1}) \leq F(\tilde{\mathbf{w}}_t) + \langle \nabla F(\tilde{\mathbf{w}}_t), \tilde{\mathbf{w}}_{t+1} - \tilde{\mathbf{w}}_t \rangle + \frac{L_F}{2} \|\tilde{\mathbf{w}}_{t+1} - \tilde{\mathbf{w}}_t\|^2.$$

Now, we use the above recursive equation

$$\tilde{\mathbf{w}}_{t+1} = \tilde{\mathbf{w}}_t - \gamma \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \nabla F_i(\mathbf{w}_t) - \tilde{T}.$$

Substituting, we obtain

$$\begin{aligned} & F(\tilde{\mathbf{w}}_{t+1}) \\ & \leq F(\tilde{\mathbf{w}}_t) - \gamma \langle \nabla F(\tilde{\mathbf{w}}_t), \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \nabla F_i(\mathbf{w}_t) \rangle - \langle \nabla F(\tilde{\mathbf{w}}_t), \tilde{T} \rangle + \frac{L_F}{2} \left\| \frac{\gamma}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \nabla F_i(\mathbf{w}_t) + \tilde{T} \right\|^2 \\ & \leq F(\tilde{\mathbf{w}}_t) - \gamma \langle \nabla F(\tilde{\mathbf{w}}_t), \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \nabla F_i(\mathbf{w}_t) \rangle - \langle \nabla F(\tilde{\mathbf{w}}_t), \tilde{T} \rangle + L_F \gamma^2 \left\| \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \nabla F_i(\mathbf{w}_t) \right\|^2 + L_F \|\tilde{T}\|^2 \end{aligned} \quad (4.11)$$

In the subsequent calculation, we use the following definition of smoothness:

$$\|\nabla F(\mathbf{y}_1) - \nabla F(\mathbf{y}_2)\| \leq L_F \|\mathbf{y}_1 - \mathbf{y}_2\|$$

for all \mathbf{y}_1 and $\mathbf{y}_2 \in \mathbb{R}^d$.

Rewriting the right hand side (R.H.S) of equation (4.11), we obtain

$$\begin{aligned} R.H.S = & \underbrace{F(\tilde{\mathbf{w}}_t) - \gamma \langle \nabla F(\tilde{\mathbf{w}}_t), \nabla F(\mathbf{w}_t) \rangle}_{Term-I} + \underbrace{\gamma \langle \nabla F(\tilde{\mathbf{w}}_t), \nabla F(\mathbf{w}_t) - \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \nabla F_i(\mathbf{w}_t) \rangle}_{Term-II} \\ & + \underbrace{\langle \nabla F(\mathbf{w}_t), -\tilde{T} \rangle + \langle \nabla F(\tilde{\mathbf{w}}_t) - \nabla F(\mathbf{w}_t), -\tilde{T} \rangle}_{Term-III} \\ & + \underbrace{2L_F \gamma^2 \left\| \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t) \right\|^2 + 2L_F \gamma^2 \|\nabla F(\mathbf{w}_t)\|^2 + L_F \|\tilde{T}\|^2}_{Term-IV}. \end{aligned}$$

We now control the 4 terms separately. We start with Term-I.

Control of Term-I: We obtain

$$\begin{aligned} \text{Term-I} &= F(\tilde{\mathbf{w}}_t) - \gamma \langle \nabla F(\mathbf{w}_t), \nabla F(\mathbf{w}_t) \rangle - \gamma \langle \nabla F(\tilde{\mathbf{w}}_t) - \nabla F(\mathbf{w}_t), \nabla F(\mathbf{w}_t) \rangle \\ &\leq F(\tilde{\mathbf{w}}_t) - \gamma \|\nabla F(\mathbf{w}_t)\|^2 + 25\gamma \|\nabla F(\tilde{\mathbf{w}}_t) - \nabla F(\mathbf{w}_t)\|^2 + \frac{\gamma}{100} \|\nabla F(\mathbf{w}_t)\|^2, \end{aligned}$$

where we use Young's inequality ($\langle a, b \rangle \leq \frac{\rho}{2} \|a\|^2 + \frac{1}{2\rho} \|b\|^2$ with $\rho = 50$) in the last inequality. Using the smoothness of $F(\cdot)$, we obtain

$$\text{Term-I} \leq F(\tilde{\mathbf{w}}_t) - \gamma \|\nabla F(\mathbf{w}_t)\|^2 + \frac{\gamma}{100} \|\nabla F(\mathbf{w}_t)\|^2 + 25\gamma L_F^2 \left\| \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathbf{e}_i(t) \right\|^2. \quad (4.12)$$

Control of Term-II: Similarly, for Term-II, we have

$$\begin{aligned} \text{Term-II} &= \gamma \langle \nabla F(\tilde{\mathbf{w}}_t), \nabla F(\mathbf{w}_t) - \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \nabla F_i(\mathbf{w}_t) \rangle \leq 50\gamma \epsilon_2^2 + \frac{\gamma}{200} \|\nabla F(\tilde{\mathbf{w}}_t)\|^2 \\ &\leq 50\gamma \epsilon_2^2 + \frac{\gamma}{100} \|\nabla F(\mathbf{w}_t)\|^2 + \frac{\gamma L_F^2}{100} \left\| \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathbf{e}_i(t) \right\|^2. \end{aligned} \quad (4.13)$$

Control of Term-III: We obtain

$$\begin{aligned} \text{Term-III} &= \langle \nabla F(\mathbf{w}_t), -\tilde{T} \rangle + \langle \nabla F(\tilde{\mathbf{w}}_t) - \nabla F(\mathbf{w}_t), -\tilde{T} \rangle \\ &\leq \frac{\gamma}{2} \|\nabla F(\mathbf{w}_t)\|^2 + \frac{1}{2\gamma} \|\tilde{T}\|^2 + \frac{L_F^2}{2} \left\| \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathbf{e}_i(t) \right\|^2 + \frac{1}{2} \|\tilde{T}\|^2. \end{aligned} \quad (4.14)$$

Control of Term-IV:

$$\begin{aligned} \text{Term-IV} &= 2L_F \gamma^2 \left\| \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t) \right\|^2 + 2L_F \gamma^2 \|\nabla F(\mathbf{w}_t)\|^2 + L_F \|\tilde{T}\|^2 \\ &\leq 2L_F \gamma^2 \epsilon_2^2 + 2L_F \gamma^2 \|\nabla F(\mathbf{w}_t)\|^2 + L_F \|\tilde{T}\|^2 \end{aligned} \quad (4.15)$$

Combining all 4 terms, we obtain

$$\begin{aligned}
F(\tilde{\mathbf{w}}_{t+1}) &\leq F(\tilde{\mathbf{w}}_t) - \left(\frac{\gamma}{2} - \frac{\gamma}{50} - 2L_F\gamma^2\right) \|\nabla F(\mathbf{w}_t)\|^2 + \left(25\gamma L_F^2 + \frac{\gamma L_F^2}{100} + \frac{L_F^2}{2}\right) \left\| \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathbf{e}_i(t) \right\|^2 \\
&\quad + 50\gamma\epsilon_2^2 + 2L_F\gamma^2\epsilon_2^2 + \left(\frac{1}{2\gamma} + \frac{1}{2} + L_F\right) \|\tilde{T}\|^2
\end{aligned} \tag{4.16}$$

We now control the error sequence and $\|\tilde{T}\|^2$. These will be separate lemmas, but here we write it as a whole.

Control of error sequence:

Lemma 4.8. *For all $i \in \mathcal{M}$, we have*

$$\|\mathbf{e}_i(t)\|^2 \leq \frac{3(1-\delta)}{\delta} \gamma^2 \sigma^2$$

for all $t \geq 0$.

Proof. For machine $i \in \mathcal{M}$, we have

$$\|\mathbf{e}_i(t+1)\|^2 = \|\mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t)) - \mathbf{p}_i(\mathbf{w}_t)\|^2 \leq (1-\delta) \|\mathbf{p}_i(\mathbf{w}_t)\|^2 = (1-\delta) \|\gamma \nabla F_i(\mathbf{w}_t) + \mathbf{e}_i(t)\|^2$$

Using technique similar to the proof of [71, Lemma 3] and using $\|\nabla F_i(\mathbf{w}_t)\|^2 \leq \sigma^2$, we obtain

$$\|\mathbf{e}_i(t+1)\|^2 \leq \frac{2(1-\delta)(1+1/\eta)}{\delta} \gamma^2 \sigma^2$$

where $\eta > 0$. Substituting $\eta = 2$ implies

$$\|\mathbf{e}_i(t+1)\|^2 \leq \frac{3(1-\delta)}{\delta} \gamma^2 \sigma^2 \tag{4.17}$$

for all $i \in \mathcal{M}$. This also implies

$$\max_{i \in \mathcal{M}} \|\mathbf{e}_i(t+1)\|^2 \leq \frac{3(1-\delta)}{\delta} \gamma^2 \sigma^2.$$

□

Control of $\|\tilde{T}\|^2$:

Lemma 4.9. *We obtain*

$$\|\tilde{T}\|^2 \leq \frac{9(1+\sqrt{1-\delta})^2 \gamma^2}{(1-\beta)^2} [\alpha^2 + \beta^2 + (\beta - \alpha)^2] \left(\epsilon_1^2 + \|\nabla F(\mathbf{w}_t)\|^2 + \frac{3(1-\delta)}{\delta} \sigma^2 \right)$$

with probability exceeding $1 - \frac{2(1-\alpha)md}{(1+n\hat{L}D)^d}$.

Proof. We have

$$\|\tilde{T}\| = \|T_1 - T_2 + T_3\| \leq \|T_1\| + \|T_2\| + \|T_3\|.$$

We control these 3 terms separately. We obtain

$$\|T_1\| = \left\| \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{B} \cap \mathcal{U}} \mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t)) \right\| \leq \frac{1}{(1-\beta)m} \sum_{i \in \mathcal{B} \cap \mathcal{U}} \|\mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t))\|.$$

Since the worker machines are sorted according to $\|\mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t))\|$ (the central machine only gets to see $\mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t))$, and so the most natural metric to sort is $\|\mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t))\|$), we obtain

$$\begin{aligned} \|T_1\| &\leq \frac{\alpha m}{(1-\beta)m} \max_{i \in \mathcal{M}} \|\mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t))\| \\ &\leq (1 + \sqrt{1-\delta}) \frac{\alpha m}{(1-\beta)m} \max_{i \in \mathcal{M}} \|\mathbf{p}_i(\mathbf{w}_t)\| \\ &\leq (1 + \sqrt{1-\delta}) \frac{\alpha m}{(1-\beta)m} \max_{i \in \mathcal{M}} \|\gamma \nabla F_i(\mathbf{w}_t) + \mathbf{e}_i(t)\| \end{aligned}$$

$$\begin{aligned}
&\leq (1 + \sqrt{1 - \delta}) \frac{\alpha}{(1 - \beta)} \gamma \max_{i \in \mathcal{M}} \|\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)\| + (1 + \sqrt{1 - \delta}) \frac{\alpha}{(1 - \beta)} \gamma \|\nabla F(\mathbf{w}_t)\| \\
&+ (1 + \sqrt{1 - \delta}) \frac{\alpha}{(1 - \beta)} \max_{i \in \mathcal{M}} \|\mathbf{e}_i(t)\| \\
&\leq (1 + \sqrt{1 - \delta}) \frac{\alpha \gamma \epsilon_1}{(1 - \beta)} + (1 + \sqrt{1 - \delta}) \frac{\alpha \gamma}{(1 - \beta)} \|\nabla F(\mathbf{w}_t)\| \\
&+ (1 + \sqrt{1 - \delta}) \frac{\alpha \gamma \sigma}{(1 - \beta)} \sqrt{\frac{3(1 - \delta)}{\delta}}.
\end{aligned}$$

Hence,

$$\|T_1\|^2 \leq 3 \frac{(1 + \sqrt{1 - \delta})^2}{(1 - \beta)^2} \alpha^2 \gamma^2 \left(\epsilon_1^2 + \|\nabla F(\mathbf{w}_t)\|^2 + \frac{3(1 - \delta)}{\delta} \sigma^2 \right).$$

Similarly, we obtain,

$$\|T_2\|^2 \leq 3 \frac{(1 + \sqrt{1 - \delta})^2}{(1 - \beta)^2} \beta^2 \gamma^2 \left(\epsilon_1^2 + \|\nabla F(\mathbf{w}_t)\|^2 + \frac{3(1 - \delta)}{\delta} \sigma^2 \right).$$

For T_3 , we have

$$\begin{aligned}
\|T_3\| &= \frac{\beta - \alpha}{1 - \beta} \left\| \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathcal{Q}(\mathbf{p}_i(\mathbf{w}_t)) \right\| \leq \frac{\beta - \alpha}{1 - \beta} \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} (1 + \sqrt{1 - \delta}) \|\mathbf{p}_i(\mathbf{w}_t)\| \\
&\leq (1 + \sqrt{1 - \delta}) \frac{\beta - \alpha}{1 - \beta} \max_{i \in \mathcal{M}} \|\mathbf{p}_i(\mathbf{w}_t)\|
\end{aligned}$$

Using the previous calculation, we obtain

$$\begin{aligned}
\|T_3\| &\leq (1 + \sqrt{1 - \delta}) \frac{(\beta - \alpha) \gamma \epsilon_1}{(1 - \beta)} + (1 + \sqrt{1 - \delta}) \frac{(\beta - \alpha) \gamma}{(1 - \beta)} \|\nabla F(\mathbf{w}_t)\| \\
&+ (1 + \sqrt{1 - \delta}) \frac{(\beta - \alpha) \gamma \sigma}{(1 - \beta)} \sqrt{\frac{3(1 - \delta)}{\delta}},
\end{aligned}$$

and as a result,

$$\|T_3\|^2 \leq 3 \frac{(1 + \sqrt{1 - \delta})^2}{(1 - \beta)^2} (\beta - \alpha)^2 \gamma^2 \left(\epsilon_1^2 + \|\nabla F(\mathbf{w}_t)\|^2 + \frac{3(1 - \delta)}{\delta} \sigma^2 \right).$$

Combining the above 3 terms, we obtain

$$\begin{aligned}\|\tilde{T}\|^2 &\leq 3\|T_1\|^2 + 3\|T_2\|^2 + 3\|T_3\|^2 \\ &\leq \frac{9(1 + \sqrt{1 - \delta})^2 \gamma^2}{(1 - \beta)^2} [\alpha^2 + \beta^2 + (\beta - \alpha)^2] \left(\epsilon_1^2 + \|\nabla F(\mathbf{w}_t)\|^2 + \frac{3(1 - \delta)}{\delta} \sigma^2 \right).\end{aligned}$$

□

Back to the convergence of $F(\cdot)$: We use the above bound on $\|\tilde{T}\|^2$ and Lemma 4.8 to conclude the proof of the main convergence result. Recall equation (4.16):

$$\begin{aligned}F(\tilde{\mathbf{w}}_{t+1}) &\leq F(\tilde{\mathbf{w}}_t) - \left(\frac{\gamma}{2} - \frac{\gamma}{50} - 2L_F\gamma^2 \right) \|\nabla F(\mathbf{w}_t)\|^2 + \left(25\gamma L_F^2 + \frac{\gamma L_F^2}{100} + \frac{L_F^2}{2} \right) \left\| \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathbf{e}_i(t) \right\|^2 \\ &\quad + 50\gamma\epsilon_2^2 + 2L_F\gamma^2\epsilon_2^2 + \left(\frac{1}{2\gamma} + \frac{1}{2} + L_F \right) \|\tilde{T}\|^2\end{aligned}$$

First, let us compute the term associated with the error sequence. Note that (from Cauchy-Schwartz inequality)

$$\left\| \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathbf{e}_i(t) \right\|^2 \leq \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \|\mathbf{e}_i(t)\|^2,$$

and from equation (4.17), we obtain

$$\left\| \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathbf{e}_i(t) \right\|^2 \leq \frac{3(1 - \delta)}{\delta} \gamma^2 \sigma^2,$$

and so the error term is upper bounded by

$$\left(\frac{\gamma^2 L_F^2}{2} + \frac{\gamma^3 L_F^2}{100} + 25\gamma^3 L_F^2 \right) \frac{3(1 - \delta) \sigma^2}{\delta}.$$

We now substitute the expression for $\|\tilde{T}\|^2$. We obtain

$$\left(\frac{1}{2\gamma} + \frac{1}{2} + L_F \right) \|\tilde{T}\|^2 = \frac{1}{2\gamma} \|\tilde{T}\|^2 + \left(\frac{1}{2} + L_F \right) \|\tilde{T}\|^2.$$

The first term in the above equation is

$$\begin{aligned}
\frac{1}{2\gamma} \|\tilde{T}\|^2 &\leq \frac{9\gamma(1+\sqrt{1-\delta})^2}{2(1-\beta)^2} [\alpha^2 + \beta^2 + (\beta - \alpha)^2] \left(\epsilon_1^2 + \|\nabla F(\mathbf{w}_t)\|^2 + \frac{3(1-\delta)}{\delta} \sigma^2 \right) \\
&\leq \frac{9\gamma(1+\sqrt{1-\delta})^2}{2(1-\beta)^2} [\alpha^2 + \beta^2 + (\beta - \alpha)^2] \|\nabla F(\mathbf{w}_t)\|^2 \\
&\quad + \frac{9\gamma(1+\sqrt{1-\delta})^2}{2(1-\beta)^2} [\alpha^2 + \beta^2 + (\beta - \alpha)^2] \left(\epsilon_1^2 + \frac{3(1-\delta)}{\delta} \sigma^2 \right),
\end{aligned}$$

and the second term is

$$\begin{aligned}
&\left(\frac{1}{2} + L_F \right) \|\tilde{T}\|^2 \\
&\leq \left(\frac{1}{2} + L_F \right) \frac{9\gamma^2(1+\sqrt{1-\delta})^2}{(1-\beta)^2} [\alpha^2 + \beta^2 + (\beta - \alpha)^2] \left(\epsilon_1^2 + \|\nabla F(\mathbf{w}_t)\|^2 + \frac{3(1-\delta)}{\delta} \sigma^2 \right) \\
&\leq \left(\frac{1}{2} + L_F \right) \frac{9\gamma^2(1+\sqrt{1-\delta})^2}{(1-\beta)^2} [\alpha^2 + \beta^2 + (\beta - \alpha)^2] \|\nabla F(\mathbf{w}_t)\|^2 \\
&\quad + \left(\frac{1}{2} + L_F \right) \frac{9\gamma^2(1+\sqrt{1-\delta})^2}{(1-\beta)^2} [\alpha^2 + \beta^2 + (\beta - \alpha)^2] \left(\epsilon_1^2 + \frac{3(1-\delta)}{\delta} \sigma^2 \right)
\end{aligned}$$

Collecting all the above terms, the coefficient of $-\gamma \|\nabla F(\mathbf{w}_t)\|^2$ is given by

$$\begin{aligned}
&\frac{1}{2} - \frac{1}{50} - 2L_F\gamma - \frac{9(1+\sqrt{1-\delta})^2}{2(1-\beta)^2} [\alpha^2 + \beta^2 + (\beta - \alpha)^2] \\
&\quad - \left(\frac{1}{2} + L_F \right) \frac{9\gamma(1+\sqrt{1-\delta})^2}{(1-\beta)^2} [\alpha^2 + \beta^2 + (\beta - \alpha)^2].
\end{aligned}$$

Provided we select a sufficiently small γ , a little algebra shows that if

$$\frac{9(1+\sqrt{1-\delta})^2}{2(1-\beta)^2} [\alpha^2 + \beta^2 + (\beta - \alpha)^2] < \left(\frac{1}{2} - \frac{1}{50} \right),$$

the coefficient of $\|\nabla F(\mathbf{w}_t)\|^2$ becomes $-c\gamma$, where $c > 0$ is a universal constant.

Considering the other terms and rewriting equation (4.16), we obtain

$$F(\tilde{\mathbf{w}}_{t+1}) \leq F(\tilde{\mathbf{w}}_t) - c\gamma \|\nabla F(\mathbf{w}_t)\|^2 + \left(\frac{\gamma^2 L_F^2}{2} + \frac{\gamma^3 L_F^2}{100} + 25\gamma^3 L_F^2 \right) \frac{3(1-\delta)\sigma^2}{\delta} + 50\gamma\epsilon_2^2$$

$$\begin{aligned}
& + 2L_F\gamma^2\epsilon_2^2 + \frac{9\gamma(1+\sqrt{1-\delta})^2}{2(1-\beta)^2} [\alpha^2 + \beta^2 + (\beta - \alpha)^2] \left(\epsilon_1^2 + \frac{3(1-\delta)}{\delta}\sigma^2 \right) \\
& + \left(\frac{1}{2} + L_F \right) \frac{9\gamma^2(1+\sqrt{1-\delta})^2}{(1-\beta)^2} [\alpha^2 + \beta^2 + (\beta - \alpha)^2] \left(\epsilon_1^2 + \frac{3(1-\delta)}{\delta}\sigma^2 \right).
\end{aligned}$$

Continuing, we get

$$\begin{aligned}
& \frac{1}{T+1} \sum_{t=0}^T \|\nabla F(\mathbf{w}_t)\|^2 \\
& \leq \frac{1}{c\gamma(T+1)} \sum_{t=0}^T (F(\tilde{\mathbf{w}}_t) - F(\tilde{\mathbf{w}}_{t+1})) + \left(\frac{\gamma L_F^2}{2} + \frac{\gamma^2 L_F^2}{100} + 25\gamma^2 L_F^2 \right) \frac{3(1-\delta)\sigma^2}{c\delta} + \frac{50}{c}\epsilon_2^2 \\
& + \frac{2L_F\gamma\epsilon_2^2}{c} + \frac{9(1+\sqrt{1-\delta})^2}{2c(1-\beta)^2} [\alpha^2 + \beta^2 + (\beta - \alpha)^2] \left(\epsilon_1^2 + \frac{3(1-\delta)}{\delta}\sigma^2 \right) \\
& + \left(\frac{1}{2} + L_F \right) \frac{9\gamma(1+\sqrt{1-\delta})^2}{c(1-\beta)^2} [\alpha^2 + \beta^2 + (\beta - \alpha)^2] \left(\epsilon_1^2 + \frac{3(1-\delta)}{\delta}\sigma^2 \right).
\end{aligned}$$

Using the telescoping sum, we obtain

$$\begin{aligned}
& \min_{t=0,\dots,T} \|\nabla F(\mathbf{w}_t)\|^2 \\
& \leq \frac{F(\mathbf{w}_0) - F^*}{c\gamma(T+1)} + \left[\frac{9(1+\sqrt{1-\delta})^2}{2c(1-\beta)^2} [\alpha^2 + \beta^2 + (\beta - \alpha)^2] \left(\epsilon_1^2 + \frac{3(1-\delta)}{\delta}\sigma^2 \right) + \frac{50}{c}\epsilon_2^2 \right] \\
& + \gamma \left[\frac{L_F^2}{2} \frac{3(1-\delta)\sigma^2}{c\delta} + \frac{2L_F\epsilon_2^2}{c} + \left(\frac{1}{2} + L_F \right) \frac{9(1+\sqrt{1-\delta})^2}{c(1-\beta)^2} [\alpha^2 + \beta^2 + (\beta - \alpha)^2] \left(\epsilon_1^2 + \frac{3(1-\delta)}{\delta}\sigma^2 \right) \right] \\
& + \gamma^2 \left[\left(\frac{L_F^2}{100} + 25L_F^2 \right) \frac{3(1-\delta)\sigma^2}{c\delta} \right]
\end{aligned}$$

Simplifying the above expression, we write

$$\min_{t=0,\dots,T} \|\nabla F(\mathbf{w}_t)\|^2 \leq \frac{F(\mathbf{w}_0) - F^*}{c\gamma(T+1)} + \Delta_1 + \gamma\Delta_2 + \gamma^2\Delta_3,$$

where the definition of Δ_1, Δ_2 and Δ_3 are immediate from the above expression. \square

Remark 4.11. (Choice of Step Size γ) Substituting $\gamma = \frac{1}{\sqrt{T+1}}$, we obtain

$$\min_{t=0,\dots,T} \|\nabla F(\mathbf{w}_t)\|^2 \leq \frac{F(\mathbf{w}_0) - F^*}{c\sqrt{T+1}} + \Delta_1 + \frac{\Delta_2}{\sqrt{T+1}} + \frac{\Delta_3}{T+1},$$

with high probability. Hence, we observe that the quantity associated with Δ_3 goes down at a considerably faster rate ($\mathcal{O}(1/T)$) than the other terms and hence can be ignored, when T is large.

Remark 4.12. Note that when no Byzantine worker machines are present, i.e., $\alpha = \beta = 0$, we obtain

$$\Delta_1 = \frac{50}{c}\epsilon_2^2, \quad \Delta_2 = \frac{L^2}{2} \frac{3(1-\delta)\sigma^2}{c\delta} + \frac{2L\epsilon_2^2}{c}, \quad \Delta_3 = \left(\frac{L^2}{100} + 25L^2\right) \frac{3(1-\delta)\sigma^2}{c\delta}.$$

Additionally, if $\delta = \Theta(1)$ (this is quite common in applications like training of neural nets, as mentioned earlier), we obtain $\Delta_2 = C(L^2\sigma^2 + L\epsilon_2^2)$, and $\Delta_3 = C_1L^2$. Substituting $\epsilon_2 = \mathcal{O}(\frac{d}{\sqrt{mn}})$ and for a fixed d , the upper bound in the above theorem is order-wise identical to that of standard SGD in a population loss minimization problem under similar setting [18],[59],[71, Remark 4].

Remark 4.13. (No compression setting) In the setting, where $\delta = 1$ (no compression), we obtain

$$\Delta_1 = \mathcal{O} \left[d^2 \left(\frac{\alpha^2}{n} + \frac{1}{mn} \right) \right],$$

and

$$\Delta_2 = \mathcal{O} \left[d^2 L \left(\frac{\alpha^2}{n} + \frac{1}{mn} \right) \right],$$

and $\Delta_3 = 0$. The statistical rate (obtained by making T sufficiently large) of the problem is Δ_1 , and this rate matches exactly to that of [135]. Hence, we could recover

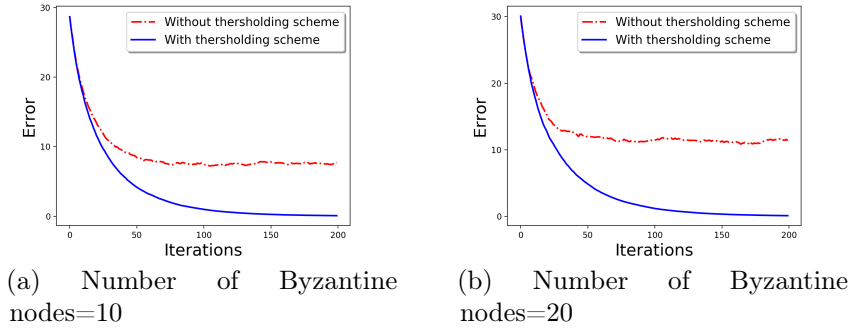


Figure 4.1: Comparison of Robust Compressed Gradient Descent with and without thresholding scheme in a regression problem. The plots show better convergence with thresholding.

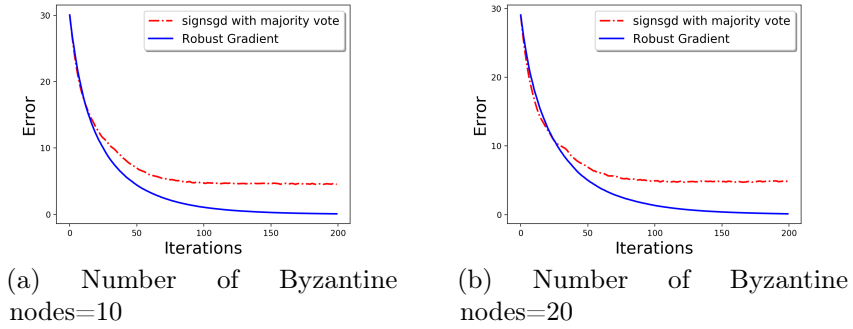


Figure 4.2: Comparison of Robust Compressed Gradient Descent with majority vote based *signSGD* [13] in regression Problem. The plots show better convergence with thresholding in comparison to the majority vote based robustness of [13]

the optimal rate without compression. Furthermore, this rate is optimal in (α, m, n) as shown in [135].

4.7 Experiments

In this section we validate the correctness of our proposed algorithms for linear regression problem and training ReLU network. In all the experiments, we choose the following compression scheme: given any $\mathbf{x} \in \mathbb{R}^d$, we report $\mathcal{C}(\mathbf{x}) = \{\frac{\|\mathbf{x}\|_1}{d}, \text{sign}(\mathbf{x})\}$ where $\text{sign}(\mathbf{x})$ serves as the quantized vector and $\frac{\|\mathbf{x}\|_1}{d}$ is the scaling factor. All the reported results are averaged over 20 different runs.

First we consider a least square regression problem $\mathbf{w}^* = \arg \min_{\mathbf{w}} \|\mathbf{A}\mathbf{w} - \mathbf{b}\|_2$. For the regression problem we generate matrix $\mathbf{A} \in \mathbb{R}^{N \times d}$, vector $\mathbf{w}^* \in \mathbb{R}^d$ by sampling each item independently from standard normal distribution and set $\mathbf{b} = \mathbf{A}\mathbf{w}^*$.

Here we choose $N = 4000$ and consider $d = 1000$. We partition the data set equally into $m = 200$ servers. We randomly choose αm ($= 10, 20$) workers to be Byzantine and apply norm based thresholding operation with parameter βm ($= 12, 22$) respectively.

We simulate the Byzantine workers by adding i.i.d $\mathcal{N}(0, 10I_d)$ entries to the gradient. In our experiments the gradient is the most pertinent information of the the worker server. So we choose to add noise to the gradient to make it a Byzantine worker. However, later on, we consider several kinds of attack models. We choose $\|\mathbf{w}_t - \mathbf{w}^*\|$ as the error metric for this problem.

Effectiveness of thresholding: We compare Algorithm 1 with compressed gradient descent (with vanilla aggregation). Our method is equipped with Byzantine tolerance steps and the vanilla compressed gradient just computes the average of the compressed gradient sent by the workers. From Figure 4.1 it is evident that the the application of norm based thresholding scheme provides better convergence result compared to the compressed gradient method without it.

Comparison with *signSGD* with majority vote: Next, in Figure 4.2, we show the comparison of our method with [13] in the regression setup described above. Our method shows a better trend in convergence.

Error-feedback with thresholding scheme: We demonstrate the effectiveness of Byzantine resilience with error-feedback scheme as described in Algorithm 2. We compare our scheme with Algorithm 1 (which does not use error feedback) in Figure 4.3. It is evident that with error-feedback, better convergence is achieved.

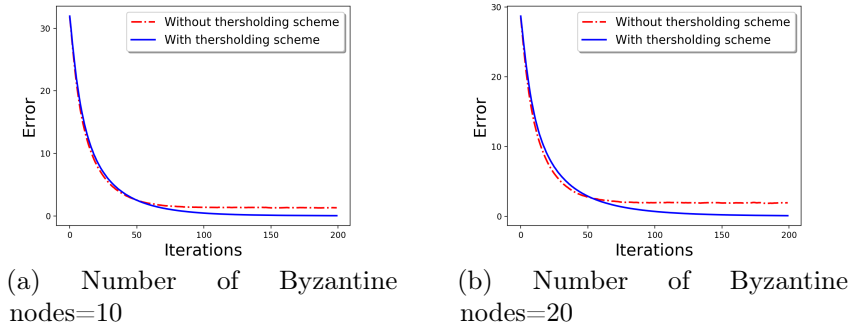


Figure 4.3: Comparison of norm based thresholding with and without error feedback. The plots show that error feedback based scheme offers better convergence.

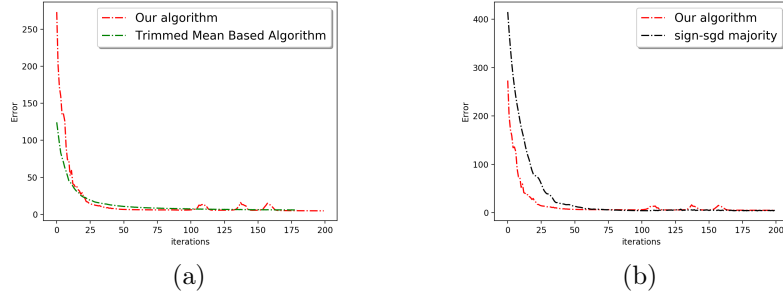


Figure 4.4: Training (cross entropy) loss for MNIST image. Comparison with (a) Uncompressed Trimmed mean [135] (b) majority based *signSGD* of [13]. In plot (a) show that Robust Gradient descent matches the convergence of the uncompressed trimmed mean [135]. Plot (b) show a faster convergence compared to the algorithm of [13].

Feed-forward Neural Net with ReLU activation: Next, we show the effectiveness of our method in training a fully connected feed forward neural net. We implement the neural net in pytorch and use the digit recognition dataset MNIST ([80]). We partition 60,000 training data into 200 different worker nodes. The neural net is equipped with 1000 node hidden layer with ReLU activation function and we choose *cross-entropy-loss* as the loss function. We simulate the Byzantine workers by adding i.i.d $\mathcal{N}(0, 10I_d)$ entries to the gradient. In Figure 4.4 we compare our robust compressed gradient descent scheme with the trimmed mean scheme of [135] and majority vote based *signSGD* scheme of [13]. Compared to the majority vote based scheme, our scheme converges faster. Further, our method shows as good

as performance of trimmed mean despite the fact the robust scheme of [135] is an uncompressed scheme and uses a more complicated aggregation rules.

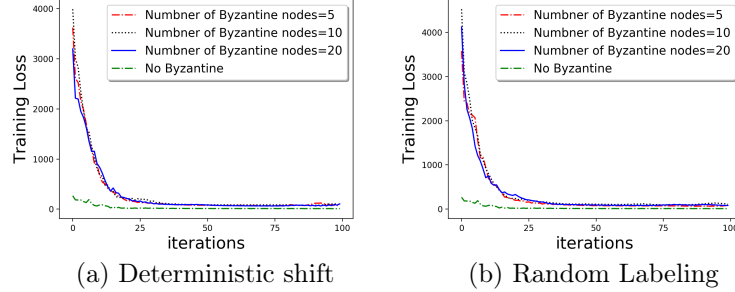


Figure 4.5: Training (cross entropy) loss for MNIST image. Different types of attack (a) labels with deterministic shift (9 – label) (b) random labels. Plots show theresholding scheme with different type of Byzantine attacks achieve similar convergence as ‘no Byzantine’ setup.

Different Types of Attacks: In the previous paragraph we compared our scheme with existing scheme with additive Gaussian noise as a form of Byzantine attack. We also show convergence results with the following type of attacks, which are quite common ([135]) in neural net training with digit recognition dataset [80]. (a) *Random label*: the Byzantine worker machines randomly replaces the labels of the data, and (b) *Deterministic Shift*: Byzantine workers in a deterministic manner replace the labels y with $9 - y$ (0 becomes 9 , 9 becomes 0). In Figure 4.5 we show the convergence with different numbers of Byzantine workers.

Large Number of Byzantine Workers: In Figures 4.6 and 4.7, we show the convergence results that holds beyond the theoretical limit (as shown in Theorem 4.4 and 4.6) of the number of Byzantine servers in the regression problem and neural net training. In Figure 4.6, for the regression problem, the Byzantine attack is additive Gaussian noise as described before and our algorithm is robust up to 40% ($\alpha = .4$) of the workers being Byzantine. While training of the feed-forward neural network, we

apply a deterministic shift as the Byzantine attack, and the algorithm converges even for 40% ($\alpha = .4$) Byzantine workers.

Another ‘natural’ Byzantine attack would be when a Byzantine worker sends $-\epsilon g$ where $0 \leq \epsilon \leq 1$ and g is the local gradient making the algorithm ‘ascent’ type. We choose $\epsilon = 0.9$ and show convergence for the regression problem for up to 40% Byzantine workers, and for the neural network training for up to 33% Byzantine workers in Figure 4.7.

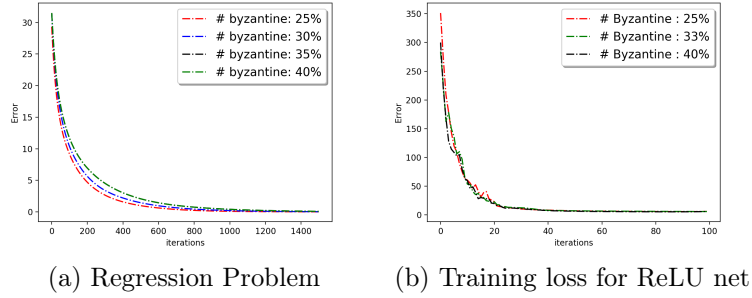


Figure 4.6: Convergence for (a) regression problem (b) training (cross entropy) loss for MNIST image. Plots show convergence beyond the theoretical bound on the number of Byzantine machine.

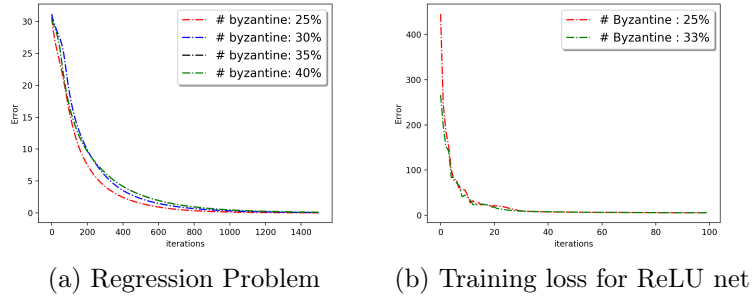


Figure 4.7: Convergence for (a) regression problem (b) training (cross entropy) loss for MNIST image. Plots show convergence with an negative Byzantine attack of $-\epsilon$ times the local gradient with high number of Byzantine machines for $\epsilon = 0.9$.

4.8 Conclusion and Future Direction

In this chapter, we address the problem of robust distributed optimization where the worker machines send the compressed gradient to the central machine. We propose

a first order optimization algorithm, and consider the settings of restricted as well as arbitrary Byzantine machines. Moreover, we consider the setup where error feedback is used to accelerate the learning process.

As a future work, it would be interesting to study the variance reduced type algorithms in this setting. In [61], the authors have studied the variance reduced gradient descent in the compressed setting. Applying similar technique with Byzantine resilience can be nice extension of the present work. Also in the Federated setup, data heterogeneity and data privacy are two very important aspect that we do not consider in this chapter. In this chapter, we consider that each coordinate of the gradient is distributed sub-exponentially as they are bounded. But these assumption can be further relaxed and extended with gradient similarity type condition (see [69]). Here worker machines send compressed gradient to the central machine, it would also be interesting to employ differential privacy in addition to compression (in Chapter 3, we achieved this).

CHAPTER 5

COMMUNICATION EFFICIENT DISTRIBUTED APPROXIMATE NEWTON METHOD

In this chapter, we propose and analyze a communication-efficient Newton-type algorithm by employing compression. The most crucial and challenging part of this work is in discerning the scope of compression in the second order optimization method. We use DINGO [32] as the baseline second order algorithm and make it communication-efficient by employing δ -approximate compressors. As mentioned, we handle one-round and two-round compression (settings 1 and 2 as described in Chapter 1) both theoretically and experimentally. We show that with proper choice of the step-size and hyper-parameters of the algorithm, we can achieve the same rate of convergence as DINGO. We prove that the gradient norm decreases exponentially over iterations of the algorithm. We also validate our results for regularized logistic regression for binary classification on real datasets[22]. Also, we emphasize here that when $\delta = 1$ (no compression), we recover the same convergence rate of DINGO[32]. Furthermore, in the regime where the compression factor δ is constant ($\Theta(1)$), with a careful choice of learning rate, our rate of convergence matches (order-wise) to that of DINGO. So we get compression for *free* in this parameter regime. Note that, as illustrated in [71], $\delta = \Theta(1)$ is usually observed in most practical applications.

Related Work

Distributed Second Order Optimization In the past few years, several distributed second order algorithms such as DANE [108], INEXACTDANE and AIDE

[99], DISCO [140] and GIANT [127] have been proposed and analyzed. These algorithms requires convexity of the objective function (in addition to 2nd order oracle access. Very recently, [102] and [32] alleviate these disadvantages. In [57], a numerical linear algebra based sketching method has been developed to compute the approximate Hessian. In this work, we deal with non-convex objective and emphasize communication efficiency.

Gradient Compression In the works [118, 125, 7, 131, 6, 12], communication efficiency is achieved by coordinate-wise quantization of gradients and in [49], vector quantization of gradients has been studied. Recently, gradient sparsification where only *top-k* component of the d -dimensional gradient vector is communicated, have been proposed in [62, 8, 3, 113]. In [71], the authors exploit the ‘error in compression’ as feedback to improve convergence of first order optimization. Very recently, Byzantine resilient communication efficient method have been analyzed in [13, 53]. Note that all the compression techniques discussed here are only applicable for first order optimization.

5.1 Background and Problem Statement

Our objective is to solve the following problem

$$\min_{\mathbf{w} \in \mathbb{R}^d} f(\mathbf{w}) = \min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^m f_i(\mathbf{w}), \quad (5.1)$$

in a distributed environment with m worker machines, where each machine has local access to the i th loss function f_i . We assume that the worker machines can communicate to the central machine, but can not interact among themselves. This is a commonly used distributed setup particularly in applications like Federated Learning, large scale neural net training etc. We assume that i th worker machine has n i.i.d data points $\{\mathbf{x}_{i,j}\}_{j=1}^n$, and hence $f_i(\mathbf{w}) = \frac{1}{n} \sum_{j=1}^n l(\mathbf{w}; \mathbf{x}_{i,j})$, where $l(\mathbf{w}; \mathbf{x}_{i,j})$ is the

loss associated with j th data point $\mathbf{x}_{i,j}$. Classically, we use the second order Newton method for convex optimization, and the update is given by,

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t \mathbf{p}_t \text{ where } \mathbf{p}_t = -[\nabla^2 f(\mathbf{w}_t)]^{-1} \nabla f(\mathbf{w}_t) \quad (5.2)$$

where α_t is the step size. Here we provide a communication efficient Newton type algorithm in distributed setup. We use the recently proposed and popular second order distributed optimization algorithm called DINGO [32].

Notation The *Moore-Penrose Inverse* of any matrix \mathbf{H} is denoted by \mathbf{H}^\dagger . For vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ by $[\mathbf{x}, \mathbf{y}]$ we denote the line $\{(1-t)\mathbf{x} + t\mathbf{y} | 0 \leq t \leq 1\}$. Also for the purpose of our algorithm we use the following definitions:

$$\begin{aligned} \mathbf{g}_{i,t} &\equiv \nabla f_i(\mathbf{w}_t) & \mathbf{H}_{i,t} &\equiv \nabla^2 f_i(\mathbf{w}_t) \\ \mathbf{g}_t &\equiv \nabla f(\mathbf{w}_t) & \mathbf{H}_t &\equiv \nabla^2 f(\mathbf{w}_t), \\ \tilde{\mathbf{H}}_{i,t} &\equiv \begin{bmatrix} \mathbf{H}_{i,t} \\ \phi I \end{bmatrix} \in \mathbb{R}^{2d \times d} & \tilde{\mathbf{g}}_{i,t} &\equiv \begin{bmatrix} \mathbf{g}_{i,t} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{2d}, \end{aligned} \quad (5.3)$$

where $\phi > 0$ and $\mathbf{0} \in \mathbb{R}^d$ is the all zero vector.

We conclude this section with the set of assumptions required for the analysis presented in the subsequent sections.

Assumption 5.1 (Twice Differentiability). *For all $i \in [m]$, the local loss functions f_i are twice differentiable.*

Assumption 5.2 (Moral Smoothness). *For all iteration t , there exists a constant $L > 0$ such that, for all $\mathbf{w} \in [\mathbf{w}_t, \mathbf{w}_t + \mathbf{p}_t]$, where \mathbf{p}_t is the update direction we have*

$$\|\nabla^2 f(\mathbf{w}) \nabla f(\mathbf{w}) - \nabla^2 f(\mathbf{w}_t) \nabla f(\mathbf{w}_t)\| \leq L \|\mathbf{w} - \mathbf{w}_t\|.$$

Note that this is a weaker assumption than the Lipschitz-ness of both gradient and Hessian, typically used in related literature [102, 32]. The assumption requires the gradient and Hessian to be Lipschitz continuous on the piece-wise linear path of the update direction. In [32, 102], more detailed discussion on this can be found. As shown in [32], we have the following useful lemma.

Lemma 5.1. *Let $\mathbf{x}, \mathbf{z} \in \mathbb{R}^d, \beta \in (0, \infty), L \in [0, \infty)$ and $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be differentiable and suppose $\mathbf{y} \in [\mathbf{x}, \mathbf{z}]$. If $\|\nabla f(\mathbf{y}) - \nabla f(\mathbf{x})\| \leq L\|\mathbf{y} - \mathbf{x}\|^\beta$, then,*

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \langle \mathbf{y} - \mathbf{x}, \nabla f(\mathbf{x}) \rangle + \frac{L}{1+\beta} \|\mathbf{y} - \mathbf{x}\|^{1+\beta}.$$

Following the Assumptions 5.1, 5.2, Lemma 5.1 with $\beta = 1$ and using the fact $\nabla(\frac{1}{2}\|\nabla f(\mathbf{w})\|^2) = \nabla^2 f(\mathbf{w})\nabla f(\mathbf{w})$, we obtain

$$\begin{aligned} \|\nabla f(\mathbf{w})\|^2 &\leq \|\nabla f(\mathbf{w}_t)\|^2 + \langle \mathbf{w} - \mathbf{w}_t, \nabla^2 f(\mathbf{w}_t)\nabla f(\mathbf{w}_t) \rangle \\ &\quad + L\|\mathbf{w} - \mathbf{w}_t\|^2, \end{aligned} \tag{5.4}$$

for all $\mathbf{w} \in [\mathbf{w}_t, \mathbf{w}_t + \mathbf{p}_t]$ and all iteration t .

Assumption 5.3. *For all $i \in [m]$ there exists constants $\gamma_i \in (0, \infty)$ such that $\|\mathbf{H}_{i,t}^\dagger\| \leq \gamma_i$.*

Assumption 5.4. *For all $i \in [m]$ there exists constants $\tau_i \in (0, \infty)$ such that $\|\mathbf{H}_{i,t}\| \leq \tau_i$.*

Assumptions 5.3, 5.4 characterize the spectrum of the Hessian and its pseudo-inverse. Assumption 5.4 implies that each local Hessian has largest singular value uniformly bounded for all iterates. Also, Assumption 5.3 deals with the smallest singular value of the Hessian. If the function is strongly convex, then the smallest singular value of the Hessian is always positive. But in more general sense, the Hessian is always

positive definite in its own range space [102]. In the following assumption, we state the more general form of the Assumption 5.3.

Assumption 5.5. *There exists an constant $\beta \in (0, \infty)$ such that for all iteration t we have $\|\mathbf{H}_t \mathbf{p}\| \geq \beta \|\mathbf{p}\|$ where \mathbf{p} is in the range space of \mathbf{H}_t i.e $\mathbf{p} \in \mathcal{R}(\mathbf{H}_t)$.*

Also an assumption similar to below appear in [32].

Assumption 5.6. *There exists a constant η_i such that $\|(\tilde{\mathbf{H}}_{i,t}^T)^\dagger \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)\| \geq \eta_i \|\mathbf{g}_t\|$.*

Observe that the assumption is dependent on the compression \mathcal{Q} . In Lemma 5.3, we justify this assumption by providing a proper value to η_i .

The next assumption is basically a restatement of Pythagoras theorem, the proof of which can be found in [102].

Assumption 5.7. *There exists a constant $\nu \in (0, 1)$ such that*

$$\|(\mathbf{U}_{\mathbf{w}}^\perp)^T \nabla f(\mathbf{w})\|^2 \leq \frac{1-\nu}{\nu} \|(\mathbf{U}_{\mathbf{w}}^T \nabla f(\mathbf{w}))\|^2,$$

for all $\mathbf{w} \in \mathbb{R}^d$, where \mathbf{U} and \mathbf{U}^\perp are the orthonormal basis for the range space of Hessian and its orthogonal complement.

Using Assumption 5.7, we infer that $\|\nabla f(\mathbf{w})\|^2 \leq \frac{1}{\nu} \|(\mathbf{U}_{\mathbf{w}}^T \nabla f(\mathbf{w}))\|^2$ for all $\mathbf{w} \in \mathbb{R}^d$.

Lemma 5.2. *If \mathcal{Q} is a δ -compressor on set $S \in \mathbb{R}^d$ as defined in definition 4.4 then*

$$\|\mathcal{Q}(\mathbf{x})\| \geq (1 - \sqrt{1 - \delta}) \|\mathbf{x}\|^2 \quad (5.5)$$

for all $\mathbf{x} \in S$ and $\delta \in [0, 1]$.

Proof. For a δ -compressor \mathcal{Q} , we have

$$\|\mathcal{Q}(\mathbf{x}) - \mathbf{x}\| \geq ||\mathcal{Q}(\mathbf{x})\| - \|\mathbf{x}\| \geq \|\mathcal{Q}(\mathbf{x})\| - \|\mathbf{x}\|$$

and we also have

$$\|\mathcal{Q}(\mathbf{x}) - \mathbf{x}\| \geq ||\mathbf{x}\| - \|\mathcal{Q}(\mathbf{x})\| \geq \|\mathcal{Q}(\mathbf{x})\| + \|\mathbf{x}\|$$

So

$$\begin{aligned} \|\mathcal{Q}(\mathbf{x})\| &\geq \|\mathbf{x}\| - \|\mathcal{Q}(\mathbf{x}) - \mathbf{x}\| \\ &\geq \|\mathbf{x}\| - \sqrt{1 - \delta}\|\mathbf{x}\| && \text{(Using the definition 4.4)} \\ &= (1 - \sqrt{1 - \delta})\|\mathbf{x}\| \end{aligned}$$

□

Lemma 5.3. *Under the Assumptions 5.4, 5.5 and 5.7, Assumption 5.6 holds with*

$$\eta_i = \beta(1 - \sqrt{1 - \delta})\left(\frac{\nu}{\tau_i^2 + \phi^2}\right)^{1/2},$$

where ϕ is described in 5.3 and δ is the compression factor.

Proof. Proof of lemma 5.3: In this proof of lemma 5.3 we are going to validate the assumption 5.6. The positive definite matrix $\tilde{\mathbf{H}}_{i,t}^T \tilde{\mathbf{H}}_{i,t}$ has eigenvalue at most $\tau_i^2 + \phi^2$.

So we have

$$\|(\tilde{\mathbf{H}}_{i,t}^T)^\dagger \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)\|^2 = \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)^T (\tilde{\mathbf{H}}_{i,t}^T \tilde{\mathbf{H}}_{i,t})^{-1} \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t) \geq \frac{1}{\tau_i^2 + \phi^2} \|\mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)\|^2$$

Now we can bound the $\|\mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)\|$ the following way

$$\|\mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)\| \geq (1 - \sqrt{1 - \delta}) \|\mathbf{H}_t \mathbf{g}_t\| \quad \text{by equation 5.5}$$

Now assume that \mathbf{U} and \mathbf{U}^\perp are the ortho-normal basis and the orthogonal complement for \mathbf{g}_t . Now we can say

$$\begin{aligned} \|\mathbf{H}_t(\mathbf{U}^T \mathbf{U} + (\mathbf{U}^\perp)^T \mathbf{U}^\perp) \mathbf{g}_t\| &\geq \|\mathbf{H}_t \mathbf{U}^T \mathbf{U} \mathbf{g}_t\| \\ &\geq \beta \|\mathbf{U}^T \mathbf{U} \mathbf{g}_t\| && \text{Using assumption 5.5} \\ &\geq \beta \sqrt{\nu} \|\mathbf{g}_t\| && \text{Using assumption 5.7} \end{aligned}$$

Finally we have

$$\|(\tilde{\mathbf{H}}_{i,t}^T)^\dagger \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)\| \geq \beta(1 - \sqrt{1 - \delta}) \left(\frac{\nu}{\tau_i^2 + \phi^2} \right)^{1/2} \|\mathbf{g}_t\|$$

□

5.2 One Round Compression

In this section, we propose and analyze an algorithm for the communication efficient second order optimization. It is formally written in Algorithm 3. The algorithm works mainly on the estimation of the gradient in the first round and the estimation of the update direction on the next. In this section, we assume that the worker machines do not compress the local gradients in the first round, i.e., $\mathcal{Q}_1(\mathbf{x}) = \mathbf{x}$ for all \mathbf{x} , in Algorithm 3. So, the central machine computes the full gradient $\mathbf{g}_t = \frac{1}{m} \sum_{i=1}^m \mathbf{g}_{i,t}$, after receiving the local gradients. Next the central machine broadcasts the gradient \mathbf{g}_t and each worker machine computes the following (compressed) vectors $\mathcal{Q}(\mathbf{H}_{i,t} \mathbf{g}_t)$, $\mathcal{Q}(\mathbf{H}_{i,t}^\dagger \mathbf{g}_t)$, $\mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger \mathbf{g}_t)$. Update direction \mathbf{p}_t and step-size α are computed based on these vectors. The iterated update $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \mathbf{p}_t$

is then performed based on the three cases (Algorithm 3). The constant \tilde{G} in the algorithm is $\theta\|\mathbf{g}_t\|^2$. We now give results on the three cases of the algorithm. For shorthand, we define $\gamma = \frac{1}{m} \sum_{i=1}^m \gamma_i$ and $\tau = \frac{1}{m} \sum_{i=1}^m \tau_i$, which are used in the results of this and subsequent sections.

Case 1 If

$$\left\langle \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t}^\dagger \mathbf{g}_t), \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t) \right\rangle \geq \theta\|\mathbf{g}_t\|^2 \quad (5.6)$$

then the update is $\mathbf{p}_t = \frac{1}{m} \sum_{i=1}^m \mathbf{p}_{i,t}$ where $\mathbf{p}_{i,t} = -\mathcal{Q}(\mathbf{H}_{i,t}^\dagger \mathbf{g}_t)$ and set $\alpha \leq \frac{1}{L\gamma(1+\sqrt{1-\delta})^2} \left[\frac{\theta(1-\rho)}{\gamma} - (1-\delta + \sqrt{1-\delta})\tau \right]$.

Theorem 5.4. *Under the Assumptions 5.1, 5.2, 5.3 and 5.4, if we run Algorithm 3 we have*

$$\|\mathbf{g}_{t+1}\|^2 \leq (1 - 2\rho\alpha\theta)\|\mathbf{g}_t\|^2. \quad (5.7)$$

Proof. Proof of Theorem 5.4 Following from the equation 5.4 and with update $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha\mathbf{p}_t$ we have

$$\|\mathbf{g}_{t+1}\|^2 \leq \|\mathbf{g}_t\|^2 + 2\alpha\langle \mathbf{p}_t, \mathbf{H}_t \mathbf{g}_t \rangle + L\alpha^2\|\mathbf{p}_t\|^2 \quad (5.8)$$

First we are going to compute the bound on \mathbf{p}_t where $\mathbf{p}_t = -\frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t)$

$$\begin{aligned} \|\mathbf{p}_t\| &= \left\| \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t}^\dagger \mathbf{g}_t) \right\| \\ &\leq \frac{1}{m} \sum_{i=1}^m (1 + \sqrt{1-\delta}) \|\mathbf{H}_{i,t}^\dagger \mathbf{g}_t\| && \text{(Definition of compressor)} \\ &\leq \frac{1}{m} \sum_{i=1}^m (1 + \sqrt{1-\delta}) \gamma_i \|\mathbf{g}_t\| && \text{Form assumption 5.3} \\ &= (1 + \sqrt{1-\delta}) \gamma \|\mathbf{g}_t\| && \left(\frac{1}{m} \sum_{i=1}^m \gamma_i = \gamma \right) \end{aligned}$$

Now we control the cross product term

$$\langle \mathbf{p}_t, \mathbf{H}_t \mathbf{g}_t \rangle = \underbrace{\langle \mathbf{p}_t, \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t} \mathbf{g}_t) \rangle}_{Term1} + \underbrace{\langle \mathbf{p}_t, \frac{1}{m} \sum_{i=1}^m \mathbf{H}_{i,t} \mathbf{g}_t - \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t} \mathbf{g}_t) \rangle}_{Term2} \quad (5.9)$$

Now the Term 1 can be bounded by the condition 5.6

$$\langle \mathbf{p}_t, \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t} \mathbf{g}_t) \rangle = \langle -\frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t), \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t} \mathbf{g}_t) \rangle \leq -\theta \|\mathbf{g}_t\|^2$$

We bound the Term 2

$$\begin{aligned} & \langle \mathbf{p}_t, \frac{1}{m} \sum_{i=1}^m \mathbf{H}_{i,t} \mathbf{g}_t - \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t} \mathbf{g}_t) \rangle \\ & \leq \|\mathbf{p}_t\| \left\| \frac{1}{m} \sum_{i=1}^m \mathbf{H}_{i,t} \mathbf{g}_t - \mathcal{Q}(\mathbf{H}_{i,t} \mathbf{g}_t) \right\| \\ & \leq \|\mathbf{p}_t\| \left(\frac{1}{m} \sum_{i=1}^m \sqrt{1-\delta} \|\mathbf{H}_{i,t} \mathbf{g}_t\| \right) \quad (\delta - \text{compressor}) \\ & \leq \|\mathbf{p}_t\| \left(\frac{1}{m} \sum_{i=1}^m \sqrt{1-\delta} \tau_i \|\mathbf{g}_t\| \right) \quad (\text{Assumption 5.4}) \\ & \leq \left(\frac{1}{m} \sum_{i=1}^m (1 + \sqrt{1-\delta}) \gamma_i \right) \left(\frac{1}{m} \sum_{i=1}^m \sqrt{1-\delta} \tau_i \right) \|\mathbf{g}_t\|^2 \quad (\text{Assumption 5.3}) \\ & \leq (1 - \delta + \sqrt{1-\delta}) \gamma \tau \|\mathbf{g}_t\|^2 \quad \left(\frac{1}{m} \sum_{i=1}^m \tau_i = \tau \right) \end{aligned}$$

Now we use the bound of Term 1 and Term 2 and put it in equation 5.9

$$\langle \mathbf{p}_t, \mathbf{H}_t \mathbf{g}_t \rangle \leq (-\theta + (1 - \delta + \sqrt{1-\delta}) \gamma \tau) \|\mathbf{g}_t\|^2$$

Collecting all the terms and plugging in equation 5.8, we have

$$\|\mathbf{g}_{t+1}\|^2 \leq \|\mathbf{g}_t\|^2 + 2\alpha(-\theta + (1 - \delta + \sqrt{1-\delta}) \gamma \tau) \|\mathbf{g}_t\|^2 + L\alpha^2 \gamma^2 (1 + \sqrt{1-\delta})^2 \|\mathbf{g}_t\|^2$$

$$\begin{aligned}
&\leq \|\mathbf{g}_t\|^2 + 2\alpha(-\theta + (1 - \delta + \sqrt{1 - \delta})\gamma\tau + \frac{L\alpha\gamma^2}{2}(1 + \sqrt{1 - \delta})^2)\|\mathbf{g}_t\|^2 \\
&\leq (1 - 2\alpha\rho\theta)\|\mathbf{g}_t\|^2
\end{aligned}$$

where $\theta(1 - \rho) = (1 - \delta + \sqrt{1 - \delta})\gamma\tau + \frac{L\alpha\gamma^2}{2}(1 + \sqrt{1 - \delta})^2$. Simplifying it we get

$$\alpha = \frac{1}{L\gamma(1 + \sqrt{1 - \delta})^2} \left[\frac{\theta(1 - \rho)}{\gamma} - (1 - \delta + \sqrt{1 - \delta})\tau \right]$$

□

Remark 5.1. *Note that we achieve an exponential convergence even for non-convex functions. When δ is a constant, our algorithm enjoys the same order of convergence with compression as in [71]. So, we get the compression for free.*

Remark 5.2. *Case 1 is often satisfied if we choose the value of the hyper-parameter $\theta \sim \gamma\tau$ and a constant δ . Experimentally we find that it to be the most frequent case.*

Case 2 If

$$\left\langle \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t), \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t) \right\rangle \geq \theta \|\mathbf{g}_t\|^2 \quad (5.10)$$

then the update is $\mathbf{p}_t = \frac{1}{m} \sum_{i=1}^m \mathbf{p}_{i,t}$ where $\mathbf{p}_{i,t} = -\mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t)$ and set $\alpha \leq \frac{2\phi^2}{L((1+\sqrt{1-\delta})^2)} [\theta(1 - \rho) - (1 - \delta + \sqrt{1 - \delta})\frac{\tau}{\phi}]$. Here, (from the construction) we assume

$$\|\tilde{\mathbf{H}}_{i,t}^\dagger\| \leq 1/\phi. \quad (5.11)$$

Theorem 5.5. *Under the Assumptions 5.1, 5.2, 5.4 and condition (5.11), if we run Algorithm 3 we have*

$$\|\mathbf{g}_{t+1}\|^2 \leq (1 - 2\rho\alpha\theta)\|\mathbf{g}_t\|^2. \quad (5.12)$$

Proof. Proof of Theorem 5.5 Similarly to the proof of 5.4, we bound the term \mathbf{p}_t for the case 2 where $\mathbf{p}_t = -\frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t)$.

$$\begin{aligned}
\|\mathbf{p}_t\| &= \left\| \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t) \right\| \\
&\leq \frac{1}{m} \sum_{i=1}^m (1 + \sqrt{1 - \delta}) \|\tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t\| && (\delta\text{-Compressor}) \\
&\leq \frac{1}{m} \sum_{i=1}^m (1 + \sqrt{1 - \delta}) \frac{1}{\phi} \|\mathbf{g}_t\| && (\text{From equation 5.11}) \\
&= (1 + \sqrt{1 - \delta}) \frac{1}{\phi} \|\mathbf{g}_t\|
\end{aligned}$$

Now we control the dot product term

$$\langle \mathbf{p}_t, \mathbf{H}_t \mathbf{g}_t \rangle = \underbrace{\langle \mathbf{p}_t, \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t} \mathbf{g}_t) \rangle}_{Term1} + \underbrace{\langle \mathbf{p}_t, \frac{1}{m} \sum_{i=1}^m \mathbf{H}_{i,t} \mathbf{g}_t - \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t} \mathbf{g}_t) \rangle}_{Term2} \quad (5.13)$$

Now the Term 1 can be bounded by the condition 5.10

$$\langle \mathbf{p}_t, \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t} \mathbf{g}_t) \rangle = \langle -\frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t), \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t} \mathbf{g}_t) \rangle \leq -\theta \|\mathbf{g}_t\|^2$$

We bound the Term 2

$$\begin{aligned}
&\langle \mathbf{p}_t, \frac{1}{m} \sum_{i=1}^m \mathbf{H}_{i,t} \mathbf{g}_t - \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t} \mathbf{g}_t) \rangle \\
&\leq \|\mathbf{p}_t\| \left\| \frac{1}{m} \sum_{i=1}^m \mathbf{H}_{i,t} \mathbf{g}_t - \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t} \mathbf{g}_t) \right\| \\
&\leq \|\mathbf{p}_t\| \left(\frac{1}{m} \sum_{i=1}^m \sqrt{1 - \delta} \tau_i \|\mathbf{g}_t\| \right) && (\delta\text{-Compressor}) \\
&\leq ((1 + \sqrt{1 - \delta}) \frac{1}{\phi}) \left(\frac{1}{m} \sum_{i=1}^m \sqrt{1 - \delta} \tau_i \right) \|\mathbf{g}_t\|^2 \\
&= (1 - \delta + \sqrt{1 - \delta}) \frac{\tau}{\phi} \|\mathbf{g}_t\|^2 && (\frac{1}{m} \sum_{i=1}^m \tau_i = \tau)
\end{aligned}$$

So we have the following bound for equation 5.13

$$\langle \mathbf{p}_t, \mathbf{H}_t \mathbf{g}_t \rangle = (-\theta + (1 - \delta + \sqrt{1 - \delta}) \frac{\tau}{\phi}) \|\mathbf{g}_t\|^2$$

Collecting all the terms and plugging in equation 5.8, we have

$$\begin{aligned} \|\mathbf{g}_{t+1}\|^2 &\leq \|\mathbf{g}_t\|^2 + 2\alpha(-\theta + (1 - \delta + \sqrt{1 - \delta}) \frac{\tau}{\phi}) \|\mathbf{g}_t\|^2 + L\alpha^2((1 + \sqrt{1 - \delta}) \frac{1}{\phi})^2 \|\mathbf{g}_t\|^2 \\ &= \|\mathbf{g}_t\|^2 + 2\alpha(-\theta + (1 - \delta + \sqrt{1 - \delta}) \frac{\tau}{\phi} + L\alpha((1 + \sqrt{1 - \delta})^2 \frac{1}{2\phi^2})) \|\mathbf{g}_t\|^2 \\ &\leq (1 - 2\alpha\rho\theta) \|\mathbf{g}_t\|^2 \end{aligned}$$

here $\theta(1 - \rho) = (1 - \delta + \sqrt{1 - \delta}) \frac{\tau}{\phi} + L\alpha((1 + \sqrt{1 - \delta})^2 \frac{1}{2\phi^2})$. We get the following

$$\alpha = \frac{2\phi^2}{L((1 + \sqrt{1 - \delta})^2)} [\theta(1 - \rho) - (1 - \delta + \sqrt{1 - \delta}) \frac{\tau}{\phi}]$$

□

Remark 5.3. *We resort to case 2 if the condition for case 1 is not satisfied. We observe the similar exponential convergence. The convergence rate here depends on the choice of ϕ which is the spectral upper bound of $\tilde{\mathbf{H}}^\dagger$. In our experiments, we did not encounter this case.*

Case 3 When the conditions for case 1 and case 2 are not satisfied, the central machine broadcasts $\mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)$ and solve the following optimization problem locally:

$$\begin{aligned} &\operatorname{argmin}_{\mathbf{p}} \|\tilde{\mathbf{H}}_{i,t} \mathbf{p} + \tilde{\mathbf{g}}_t\|^2 \\ &\text{such that } \langle \mathbf{p}, \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t) \rangle \leq -\theta \|\mathbf{g}_t\|^2. \end{aligned} \tag{5.14}$$

Proposition 5.6. *The solution to the optimization problem (5.14) is*

$$\hat{\mathbf{p}}_{i,t} = -\tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t - \lambda_{i,t} (\tilde{\mathbf{H}}_{i,t}^T \tilde{\mathbf{H}}_{i,t})^{-1} \mathcal{Q}(\mathbf{H}_t \hat{\mathbf{g}}_t)$$

$$\text{where } \lambda_{i,t} = \frac{\theta \|\hat{\mathbf{g}}_t\|^2 - (\mathcal{Q}(\mathbf{H}_t \hat{\mathbf{g}}_t))^T \tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t}{(\mathcal{Q}(\mathbf{H}_t \hat{\mathbf{g}}_t))^T (\tilde{\mathbf{H}}_{i,t}^T \tilde{\mathbf{H}}_{i,t})^{-1} \mathcal{Q}(\mathbf{H}_t \hat{\mathbf{g}}_t)}.$$

In Algorithm 3, we use $\mathbf{p}_{i,t}$ (to get the direction \mathbf{p}_t), which is defined as: $\mathbf{p}_{i,t} = \mathcal{Q}(\hat{\mathbf{p}}_{i,t})$ and set $\alpha \leq \frac{2}{L(1+\sqrt{1-\delta})^2 c^2} (\theta(1-\rho) - (2\sqrt{1-\delta}c\tau))$, where $c \equiv \frac{1}{m} \sum_{i=1}^m (\frac{1}{\phi}(2 + \frac{\theta}{\eta_i}))$.

Theorem 5.7. *Suppose Assumptions 5.1, 5.6 and Lemma 5.3 hold. Then, Algorithm 3 yields*

$$\|\mathbf{g}_{t+1}\|^2 \leq (1 - 2\rho\alpha\theta) \|\mathbf{g}_t\|^2. \quad (5.15)$$

Proof. Proof of Theorem 5.7 The update here is $\mathbf{p}_t = \frac{1}{m} \sum_{i=1}^m \mathbf{p}_{i,t}$ where

$$\mathbf{p}_{i,t} = -\mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t - \lambda_{i,t} (\tilde{\mathbf{H}}_{i,t}^T \tilde{\mathbf{H}}_{i,t})^{-1} \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)) \text{ where } \lambda_{i,t} = \frac{\theta \|\mathbf{g}_t\|^2 - (\mathcal{Q}(\mathbf{H}_t \mathbf{g}_t))^T \tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t}{(\mathcal{Q}(\mathbf{H}_t \mathbf{g}_t))^T (\tilde{\mathbf{H}}_{i,t}^T \tilde{\mathbf{H}}_{i,t})^{-1} \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)} \quad (5.16)$$

First we bound \mathbf{p}_t

$$\begin{aligned} \|\mathbf{p}_t\| &= \left\| \frac{1}{m} \sum_{i=1}^m \mathbf{p}_{i,t} \right\| \\ &\leq \frac{1}{m} \sum_{i=1}^m \|\mathbf{p}_{i,t}\| \\ &\leq (1 + \sqrt{1+\delta}) \frac{1}{m} \sum_{i=1}^m \|\tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t - \lambda_{i,t} (\tilde{\mathbf{H}}_{i,t}^T \tilde{\mathbf{H}}_{i,t})^{-1} \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)\| \\ &\leq (1 + \sqrt{1+\delta}) \frac{1}{m} \sum_{i=1}^m (\|\tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t\| + \|\lambda_{i,t} (\tilde{\mathbf{H}}_{i,t}^T \tilde{\mathbf{H}}_{i,t})^{-1} \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)\|) \end{aligned}$$

Now we use the fact that $(\mathcal{Q}(\mathbf{H}_t \mathbf{g}_t))^T (\tilde{\mathbf{H}}_{i,t}^T \tilde{\mathbf{H}}_{i,t})^{-1} \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t) = \|(\tilde{\mathbf{H}}_{i,t}^T)^\dagger \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)\|^2$ and bound the following term

$$\begin{aligned}
& \|\lambda_{i,t}(\tilde{\mathbf{H}}_{i,t}^T \tilde{\mathbf{H}}_{i,t})^{-1} \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)\| \\
&= \left\| \frac{\theta \|\mathbf{g}_t\|^2 - (\mathcal{Q}(\mathbf{H}_t \mathbf{g}_t))^T \tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t}{(\mathcal{Q}(\mathbf{H}_t \mathbf{g}_t))^T (\tilde{\mathbf{H}}_{i,t}^T \tilde{\mathbf{H}}_{i,t})^{-1} \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)} \right\| \|(\tilde{\mathbf{H}}_{i,t}^T \tilde{\mathbf{H}}_{i,t})^{-1} \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)\| \\
&= \left\| \frac{\theta \|\mathbf{g}_t\|^2 - (\mathcal{Q}(\mathbf{H}_t \mathbf{g}_t))^T \tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t}{\|(\tilde{\mathbf{H}}_{i,t}^T)^\dagger \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)\|^2} \right\| \|(\tilde{\mathbf{H}}_{i,t}^\dagger (\tilde{\mathbf{H}}^T)_{i,t})^\dagger \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)\| \\
&\leq \left\| \frac{\theta \|\mathbf{g}_t\|^2 - (\mathcal{Q}(\mathbf{H}_t \mathbf{g}_t))^T \tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t}{\|(\tilde{\mathbf{H}}_{i,t}^T)^\dagger \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)\|} \right\| \frac{\|(\tilde{\mathbf{H}}_{i,t}^\dagger)^\dagger \| \|(\tilde{\mathbf{H}}^T)_{i,t}\|^\dagger \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)\|}{\|(\tilde{\mathbf{H}}_{i,t}^T)^\dagger \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)\|} \\
&\leq \frac{1}{\phi} \left(\frac{\theta \|\mathbf{g}_t\|^2}{\|(\tilde{\mathbf{H}}_{i,t}^T)^\dagger \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)\|} + \frac{\|(\mathcal{Q}(\mathbf{H}_t \mathbf{g}_t))^T \tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t\|}{\|(\tilde{\mathbf{H}}_{i,t}^T)^\dagger \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)\|} \right) \\
&\leq \frac{1}{\phi} \left(\frac{\theta}{\eta_i} \|\mathbf{g}_t\| + \frac{\|\tilde{\mathbf{g}}_t\| \|(\tilde{\mathbf{H}}_{i,t}^T)^\dagger \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)\|}{\|(\tilde{\mathbf{H}}_{i,t}^T)^\dagger \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)\|} \right) \\
&\leq \frac{1}{\phi} \left(1 + \frac{\theta}{\eta_i} \right) \|\mathbf{g}_t\|
\end{aligned}$$

Also we can say that

$$\|\tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t\| \leq \frac{1}{\phi} \|\mathbf{g}_t\|$$

Finally we can bound

$$\|\mathbf{p}_t\| \leq (1 + \sqrt{1 + \delta}) \frac{1}{m} \sum_{i=1}^m \left(\frac{1}{\phi} \left(2 + \frac{\theta}{\eta_i} \right) \right) \|\mathbf{g}_t\| = c(1 + \sqrt{1 + \delta}) \|\mathbf{g}_t\|$$

where $c = \frac{1}{m} \sum_{i=1}^m \left(\frac{1}{\phi} \left(2 + \frac{\theta}{\eta_i} \right) \right)$

Now we take care of the cross product term. Assume that $\hat{\mathbf{p}}_{i,t}$ is the uncompressed vector of the $\mathbf{p}_{i,t}$, i.e $\mathbf{p}_{i,t} = \mathcal{Q}(\hat{\mathbf{p}}_{i,t})$ and $\hat{\mathbf{p}}_t = \frac{1}{m} \sum_{i=1}^m \hat{\mathbf{p}}_{i,t}$

$$\begin{aligned}
& \langle \mathbf{p}_t, \mathbf{H}_t \mathbf{g}_t \rangle \\
&= \langle \mathbf{p}_t - \hat{\mathbf{p}}_t, \mathbf{H}_t \mathbf{g}_t \rangle + \langle \hat{\mathbf{p}}_t, \mathbf{H}_t \mathbf{g}_t \rangle \\
&= \langle \mathbf{p}_t - \hat{\mathbf{p}}_t, \mathbf{H}_t \mathbf{g}_t \rangle + \langle \hat{\mathbf{p}}_t, \mathbf{H}_t \mathbf{g}_t - \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t) \rangle + \langle \hat{\mathbf{p}}_t, \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t) \rangle \\
&\leq \|\hat{\mathbf{p}}_t - \mathbf{p}_t\| \|\mathbf{H}_t \mathbf{g}_t\| + \|\hat{\mathbf{p}}_t\| \|\mathbf{H}_t \mathbf{g}_t - \mathcal{Q}(\mathbf{H}_t \mathbf{g}_t)\| - \theta \|\mathbf{g}_t\|^2 \\
&= \left\| \frac{1}{m} \sum_{i=1}^m (\hat{\mathbf{p}}_{i,t} - \mathbf{p}_{i,t}) \right\| \left\| \frac{1}{m} \sum_{i=1}^m \mathbf{H}_{i,t} \mathbf{g}_t \right\| + \left\| \frac{1}{m} \sum_{i=1}^m \hat{\mathbf{p}}_{i,t} \right\| \left\| \frac{1}{m} \sum_{i=1}^m (\mathbf{H}_{i,t} \mathbf{g}_t - \mathcal{Q}(\mathbf{H}_{i,t} \mathbf{g}_t)) \right\| - \theta \|\mathbf{g}_t\|^2 \\
&\leq 2(\sqrt{1-\delta})c \left(\frac{1}{m} \sum_{i=1}^m \tau_i \right) \|\mathbf{g}_t\|^2 - \theta \|\mathbf{g}_t\|^2 \\
&= (-\theta + 2(\sqrt{1-\delta})c\tau) \|\mathbf{g}_t\|^2
\end{aligned}$$

Collecting all the terms and plugging in equation 5.8, we have

$$\begin{aligned}
\|\mathbf{g}_{t+1}\|^2 &\leq \|\mathbf{g}_t\|^2 + 2\alpha(-\theta + 2(\sqrt{1-\delta})c\tau) \|\mathbf{g}_t\|^2 + L\alpha^2 c^2 (1 + \sqrt{1-\delta})^2 \|\mathbf{g}_t\|^2 \\
&= (1 - 2\alpha\rho\theta) \|\mathbf{g}_t\|^2
\end{aligned}$$

where $\theta(1 - \rho) = (2\sqrt{1-\delta})c\tau + \frac{1}{2}L\alpha(1 + \sqrt{1-\delta})^2 c^2$, we get bound on α

$$\alpha = \frac{2}{L(1 + \sqrt{1-\delta})^2 c^2} (\theta(1 - \rho) - (2\sqrt{1-\delta})c\tau)$$

□

Remark 5.4. *Note that, although we retain the same exponential rate of convergence, case 3 is not ideal both in terms of convergence rate and communication as it requires one more round of communication between the central and the worker machines. Fortunately, this case occurs rarely in practical situation (as we observe experimentally in Section 6.5).*

Algorithm 3

```

1: Input: Initial iterate  $\mathbf{w}_0 \in \mathbb{R}^d$ , gradient tolerance  $\xi > 0$ , Maximum iteration  $T$ ,
   parameter  $\rho \in [0, 1]$ , parameter  $\theta > 0$  and regularization parameter  $\phi > 0$  and
   Compressors  $\mathcal{Q}, \mathcal{Q}_1$ 
2: for  $t = 0, 1, \dots, T - 1$  do
3:   All worker  $i \in [m]$  locally compute and compress  $\mathcal{Q}_1(\mathbf{g}_{i,t})$  and communicate it
   to the central server
4:   Central machine compute full gradient  $\hat{\mathbf{g}}_t = \frac{1}{m} \sum_{i=1}^m \mathcal{Q}_1(\mathbf{g}_{i,t})$ 
5:   if  $\|\hat{\mathbf{g}}_t\| < \xi$  then
6:     return  $\mathbf{w}_t$ 
7:   else
8:     The central machine broadcasts  $\hat{\mathbf{g}}_t$  and in parallel each worker computes using
     compression scheme  $\mathcal{Q}(\mathbf{H}_{i,t}\hat{\mathbf{g}}_t)$ ,  $\mathcal{Q}(\mathbf{H}_{i,t}^\dagger\hat{\mathbf{g}}_t)$ ,  $\mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger\hat{\mathbf{g}}_t)$ 
9:     Central machine computes  $\mathcal{Q}(\mathbf{H}_t\hat{\mathbf{g}}_t) = \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t}\hat{\mathbf{g}}_t)$ ,  $\mathcal{Q}(\mathbf{H}_{i,t}^\dagger\hat{\mathbf{g}}_t)$ ,
      $\frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t}^\dagger\hat{\mathbf{g}}_t)$  and  $\frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger\hat{\mathbf{g}}_t)$ 
10:    if (Case 1)  $\langle \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t}^\dagger\hat{\mathbf{g}}_t), \mathcal{Q}(\mathbf{H}_t\hat{\mathbf{g}}_t) \rangle \geq \tilde{G}$  then
11:       $\mathbf{p}_t = \frac{1}{m} \sum_{i=1}^m \mathbf{p}_{i,t}$  with  $\mathbf{p}_{i,t} = -\mathcal{Q}(\mathbf{H}_{i,t}^\dagger\hat{\mathbf{g}}_t)$ 
12:    else if (Case 2)  $\langle \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger\hat{\mathbf{g}}_t), \mathcal{Q}(\mathbf{H}_t\hat{\mathbf{g}}_t) \rangle \geq \tilde{G}$  then
13:       $\mathbf{p}_t = \frac{1}{m} \sum_{i=1}^m \mathbf{p}_{i,t}$  with  $\mathbf{p}_{i,t} = -\mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger\hat{\mathbf{g}}_t)$ 
14:    else
15:      The central machine broadcasts  $\mathcal{Q}(\mathbf{H}_t\hat{\mathbf{g}}_t)$  and all the worker solve in parallel
      
$$\mathbf{p}_{i,t} = \mathcal{Q}(-\tilde{\mathbf{H}}_{i,t}^\dagger\hat{\mathbf{g}}_t - \lambda_{i,t}(\tilde{\mathbf{H}}_{i,t}^T\tilde{\mathbf{H}}_{i,t})^{-1}\mathcal{Q}(\mathbf{H}_t\hat{\mathbf{g}}_t))$$

      where  $\lambda_{i,t} = \frac{\theta\|\hat{\mathbf{g}}_t\|^2 - (\mathcal{Q}(\mathbf{H}_t\hat{\mathbf{g}}_t))^T\tilde{\mathbf{H}}_{i,t}^\dagger\hat{\mathbf{g}}_t}{(\mathcal{Q}(\mathbf{H}_t\hat{\mathbf{g}}_t))^T(\tilde{\mathbf{H}}_{i,t}^T\tilde{\mathbf{H}}_{i,t})^{-1}\mathcal{Q}(\mathbf{H}_t\hat{\mathbf{g}}_t)}$ 
16:      Compress and send  $\mathbf{p}_{i,t}$ . The central machine computes  $\mathbf{p}_t = \frac{1}{m} \sum_{i=1}^m \mathbf{p}_{i,t}$ .
17:    end if
18:    The central machine sets the value of  $\alpha$  according to text and updates
       $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha\mathbf{p}_t$ 
19:  end if
20: end for

```

So far, we let the worker machines send uncompressed local gradients and only compress the second round. Hence, with one round compression we do not gain savings in communication order-wise. In the subsequent section, we remove this issue by employing compression in both gradient and Hessian based computation.

5.3 Two Round Compression

Here, we use compression in both rounds of communication with the central machine. In particular, the worker machines use a δ -approximate compressor to compress the gradient. Hence, each worker machine sends $\mathcal{Q}_1(\mathbf{g}_{i,t})$ to the central machine (Algorithm 3). The central machine then computes $\hat{\mathbf{g}}_t = \frac{1}{m} \sum_{i=1}^m \mathcal{Q}_1(\mathbf{g}_{i,t})$ and broadcasts it. Next, the updates are done similar to the one round compression. Here, for simplicity we choose the same compression factor δ for both rounds, i.e., $\mathcal{Q}_1 = \mathcal{Q}$. Also, we choose $\tilde{G} = \theta \frac{1}{m} \sum_{i=1}^m \|\mathcal{Q}(\mathbf{g}_{i,t})\|^2$. Similar to Section 5.2, we analyze case by case basis.

Case 1 If

$$\langle \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t}^\dagger \hat{\mathbf{g}}_t), \mathcal{Q}(\mathbf{H}_t \hat{\mathbf{g}}_t) \rangle \geq \theta \frac{1}{m} \sum_{i=1}^m \|\mathcal{Q}(\mathbf{g}_{i,t})\|^2 \quad (5.17)$$

then the update is $\mathbf{p}_t = \frac{1}{m} \sum_{i=1}^m \mathbf{p}_{i,t}$ where $\mathbf{p}_{i,t} = -\mathcal{Q}(\mathbf{H}_{i,t}^\dagger \hat{\mathbf{g}}_t)$ and set

$$\alpha \leq \frac{2}{\alpha L \gamma^2 (1 + \sqrt{1 - \delta})^2} \times (\theta(1 - \rho) - \sqrt{1 - \delta}(1 + \sqrt{1 - \delta})\gamma\tau(\frac{1 - \sqrt{2 - \delta}}{1 - \sqrt{1 - \delta}}))$$

Theorem 5.8. *Under Assumptions 5.1, 5.2, 5.3 and 5.4, if we run Algorithm 3, we obtain*

$$\|\mathbf{g}_{t+1}\|^2 \leq (1 - 2\rho\alpha\theta(1 - \sqrt{1 - \delta})^2)\|\mathbf{g}_t\|^2. \quad (5.18)$$

Proof. Proof of Theorem 5.8 Here the update is

$$\mathbf{p}_t = -\frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t}^\dagger \hat{\mathbf{g}}_t)$$

where $\mathbf{g}_t = \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{g}_{i,t})$. The update is done under the condition

$$\langle \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t}^\dagger \hat{\mathbf{g}}_t), \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t} \hat{\mathbf{g}}_t) \rangle \geq \theta \frac{1}{m} \sum_{i=1}^m \|\mathcal{Q}(\mathbf{g}_{i,t})\|^2$$

For the purpose of the convergence analysis we make the following calculation

$$\begin{aligned} \frac{1}{m} \sum_{i=1}^m \|\mathcal{Q}(\mathbf{g}_{i,t})\|^2 &\geq (1 - \sqrt{1 - \delta})^2 \frac{1}{m} \sum_{i=1}^m \|\mathbf{g}_{i,t}\|^2 \\ &\geq (1 - \sqrt{1 - \delta})^2 \left\| \frac{1}{m} \sum_{i=1}^m \mathbf{g}_{i,t} \right\|^2 \\ &= (1 - \sqrt{1 - \delta})^2 \|\mathbf{g}_t\|^2 \end{aligned}$$

Now we bound the update

$$\begin{aligned} \|\mathbf{p}_t\| &= \left\| \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\mathbf{H}_{i,t}^\dagger \hat{\mathbf{g}}_t) \right\| && \text{(Equation 5.5)} \\ &\leq \frac{1}{m} \sum_{i=1}^m (1 + \sqrt{1 - \delta}) \|\mathbf{H}_{i,t}^\dagger \hat{\mathbf{g}}_t\| && (\delta - \text{Compressor}) \\ &\leq \frac{1}{m} \sum_{i=1}^m (1 + \sqrt{1 - \delta}) \gamma_i \|\hat{\mathbf{g}}_t\| && \text{(Assumption 5.3)} \\ &\leq (1 + \sqrt{1 - \delta}) \gamma \|\hat{\mathbf{g}}_t\| && (5.19) \end{aligned}$$

The cross product term is

$$\begin{aligned} \langle \mathbf{p}_t, \mathbf{H}_t \mathbf{g}_t \rangle &= \langle \mathbf{p}_t, \mathbf{H}_t \mathbf{g}_t - \mathcal{Q}(\mathbf{H}_t \hat{\mathbf{g}}_t) \rangle + \langle \mathbf{p}_t, \mathcal{Q}(\mathbf{H}_t \hat{\mathbf{g}}_t) \rangle \\ &\leq \langle \mathbf{p}_t, \mathbf{H}_t \mathbf{g}_t - \mathcal{Q}(\mathbf{H}_t \hat{\mathbf{g}}_t) \rangle - \theta \frac{1}{m} \sum_{i=1}^m \|\mathcal{Q}(\mathbf{g}_{i,t})\|^2 \end{aligned}$$

Now we bound the first term as follows

$$\langle \mathbf{p}_t, \mathbf{H}_t \mathbf{g}_t - \mathcal{Q}(\mathbf{H}_t \hat{\mathbf{g}}_t) \rangle \leq \|\mathbf{p}_t\| \|\mathbf{H}_t \mathbf{g}_t - \mathcal{Q}(\mathbf{H}_t \hat{\mathbf{g}}_t)\|$$

$$\begin{aligned}
&= \|\mathbf{p}_t\| \|\mathbf{H}_t \mathbf{g}_t - \mathbf{H}_t \hat{\mathbf{g}}_t + \mathbf{H}_t \hat{\mathbf{g}}_t - \mathcal{Q}(\mathbf{H}_t \hat{\mathbf{g}}_t)\| \\
&\leq \|\mathbf{p}_t\| (\|\mathbf{H}_t \mathbf{g}_t - \mathbf{H}_t \hat{\mathbf{g}}_t\| + \|\mathbf{H}_t \hat{\mathbf{g}}_t - \mathcal{Q}(\mathbf{H}_t \hat{\mathbf{g}}_t)\|) \\
&\leq \|\mathbf{p}_t\| (\|\mathbf{H}_t\| \|\mathbf{g}_t - \hat{\mathbf{g}}_t\| + \sqrt{1-\delta} \|\mathbf{H}_t\| \|\hat{\mathbf{g}}_t\|) \\
&\leq \sqrt{1-\delta} (1 + \sqrt{1-\delta}) \gamma \tau (\|\hat{\mathbf{g}}_t\| \frac{1}{m} \sum_{i=1}^m \|\mathbf{g}_{i,t}\| + \|\hat{\mathbf{g}}_t\|^2)
\end{aligned}$$

We further bound

$$\begin{aligned}
\|\hat{\mathbf{g}}_t\| \frac{1}{m} \sum_{i=1}^m \|\mathbf{g}_{i,t}\| &\leq (\frac{1}{m} \sum_{i=1}^m \|\mathcal{Q}(\mathbf{g}_{i,t})\|) (\frac{1}{1-\sqrt{1-\delta}} \frac{1}{m} \sum_{i=1}^m \|\mathcal{Q}(\mathbf{g}_{i,t})\|) \\
&\leq \frac{1}{1-\sqrt{1-\delta}} \frac{1}{m} \sum_{i=1}^m \|\mathcal{Q}(\mathbf{g}_{i,t})\|^2
\end{aligned}$$

Also

$$\|\hat{\mathbf{g}}_t\|^2 \leq \frac{1}{m} \sum_{i=1}^m \|\mathcal{Q}(\mathbf{g}_{i,t})\|^2$$

Assume $\tilde{G} = \frac{1}{m} \sum_{i=1}^m \|\mathcal{Q}(\mathbf{g}_{i,t})\|^2$,

$$\langle \mathbf{p}_t, \mathbf{H}_t \mathbf{g}_t - \mathcal{Q}(\mathbf{H}_t \hat{\mathbf{g}}_t) \rangle \leq \sqrt{1-\delta} (1 + \sqrt{1-\delta}) \gamma \tau (\frac{1-\sqrt{2-\delta}}{1-\sqrt{1-\delta}}) \tilde{G}$$

Collecting all we have and

$$\begin{aligned}
\|\mathbf{g}_{t+1}\|^2 &\leq \|\mathbf{g}_t\|^2 + 2\alpha(-\theta \tilde{G} + \sqrt{1-\delta} (1 + \sqrt{1-\delta}) \gamma \tau (\frac{1-\sqrt{2-\delta}}{1-\sqrt{1-\delta}}) \tilde{G}) + \alpha^2 L \gamma^2 (1 + \sqrt{1-\delta})^2 \tilde{G} \\
&= \|\mathbf{g}_t\|^2 - 2\alpha \theta \tilde{G} + 2\alpha (\sqrt{1-\delta} (1 + \sqrt{1-\delta}) \gamma \tau (\frac{1-\sqrt{2-\delta}}{1-\sqrt{1-\delta}}) \tilde{G} + \frac{\alpha L \gamma^2}{2} (1 + \sqrt{1-\delta})^2 \tilde{G}) \\
&\leq \|\mathbf{g}_t\|^2 - 2\alpha \theta \rho \tilde{G} \\
&\leq (1 - 2\alpha \theta \rho (1 - \sqrt{1-\delta})^2) \|\mathbf{g}_t\|^2
\end{aligned}$$

□

Remark 5.5. Compared to the case 1 of one round compression (Theorem 5.4), the convergence rate here suffers due to the compressed gradient information. But the exponential decay still retains. For compression factor $\delta = \Theta(1)$ (constant), we do not lose order-wise performance compared to DINGO.

Remark 5.6. Remarkably, the communication cost here is extremely low as compared the one round compression (as we see in experiments as well). Here both rounds of communication from workers to the central machine are compressed.

Case 2 If

$$\langle \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t), \mathcal{Q}(\mathbf{H}_t \hat{\mathbf{g}}_t) \rangle \geq \theta \frac{1}{m} \sum_{i=1}^m \|\mathcal{Q}(\mathbf{g}_{i,t})\|^2 \quad (5.20)$$

then the update is $\mathbf{p}_t = \frac{1}{m} \sum_{i=1}^m \mathbf{p}_{i,t}$ where $\mathbf{p}_{i,t} = -\mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t)$ and set

$$\alpha \leq \frac{2\phi^2}{\alpha L(1 + \sqrt{1 - \delta})^2} \times (\theta(1 - \rho) - \sqrt{1 - \delta}(1 + \sqrt{1 - \delta}) \frac{\tau}{\phi} (\frac{1 - \sqrt{2 - \delta}}{1 - \sqrt{1 - \delta}})).$$

Theorem 5.9. Under the assumption 5.1, 5.2, 5.4 and equation (5.11), if we run Algorithm, 3 we obtain

$$\|\mathbf{g}_{t+1}\|^2 \leq (1 - 2\rho\alpha\theta(1 - \sqrt{1 - \delta})^2) \|\mathbf{g}_t\|^2. \quad (5.21)$$

Proof. Proof of Theorem 5.9

Here the update is

$$\mathbf{p}_t = \frac{1}{m} \sum_{i=1}^m -\mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t)$$

The update is done under the condition

$$\langle \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t), \mathcal{Q}(\mathbf{H}_t \hat{\mathbf{g}}_t) \rangle \geq \theta \frac{1}{m} \sum_{i=1}^m \|\mathcal{Q}(\mathbf{g}_{i,t})\|^2$$

First we bound the update term

$$\begin{aligned}
\|\mathbf{p}_t\| &= \left\| \frac{1}{m} \sum_{i=1}^m \mathcal{Q}(\tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t) \right\| \\
&\leq \frac{1}{m} \sum_{i=1}^m (1 + \sqrt{1 - \delta}) \|\tilde{\mathbf{H}}_{i,t}^\dagger \tilde{\mathbf{g}}_t\| \\
&\leq (1 + \sqrt{1 - \delta}) \frac{1}{\phi} \|\hat{\mathbf{g}}_t\|
\end{aligned}$$

Now we take care of the cross-product term with $\tilde{G} = \frac{1}{m} \sum_{i=1}^m \|\mathcal{Q}(\mathbf{g}_{i,t})\|^2$

$$\begin{aligned}
\langle \mathbf{p}_t, \mathbf{H}_t \mathbf{g}_t \rangle &= \langle \mathbf{p}_t, \mathbf{H}_t \mathbf{g}_t - \mathcal{Q}(\mathbf{H}_t \hat{\mathbf{g}}_t) \rangle + \langle \mathbf{p}_t, \mathcal{Q}(\mathbf{H}_t \hat{\mathbf{g}}_t) \rangle \\
&\leq \langle \mathbf{p}_t, \mathbf{H}_t \mathbf{g}_t - \mathcal{Q}(\mathbf{H}_t \hat{\mathbf{g}}_t) \rangle - \theta \tilde{G}
\end{aligned}$$

Along with the previous calculation we can calculate

$$\begin{aligned}
\langle \mathbf{p}_t, \mathbf{H}_t \mathbf{g}_t - \mathcal{Q}(\mathbf{H}_t \hat{\mathbf{g}}_t) \rangle &\leq \sqrt{1 - \delta} (1 + \sqrt{1 - \delta}) \frac{\tau}{\phi} \|\hat{\mathbf{g}}_t\| \left(\|\hat{\mathbf{g}}_t\| \frac{1}{m} \sum_{i=1}^m \|\mathbf{g}_{i,t}\| + \|\hat{\mathbf{g}}_t\|^2 \right) \\
&\leq \sqrt{1 - \delta} (1 + \sqrt{1 - \delta}) \frac{\tau}{\phi} \left(\frac{1 - \sqrt{2 - \delta}}{1 - \sqrt{1 - \delta}} \right) \tilde{G}
\end{aligned}$$

Collecting all the terms we get

$$\begin{aligned}
\|\mathbf{g}_{t+1}\|^2 &\leq \|\mathbf{g}_t\|^2 + 2\alpha(-\theta \tilde{G} + \sqrt{1 - \delta} (1 + \sqrt{1 - \delta}) \frac{\tau}{\phi} \left(\frac{1 - \sqrt{2 - \delta}}{1 - \sqrt{1 - \delta}} \right) \tilde{G}) + \alpha^2 L \frac{1}{\phi^2} (1 + \sqrt{1 - \delta})^2 \tilde{G} \\
&= \|\mathbf{g}_t\|^2 - 2\alpha\theta \tilde{G} + 2\alpha(\sqrt{1 - \delta} (1 + \sqrt{1 - \delta}) \frac{\tau}{\phi} \left(\frac{1 - \sqrt{2 - \delta}}{1 - \sqrt{1 - \delta}} \right) \tilde{G} + \frac{\alpha L}{2\phi^2} (1 + \sqrt{1 - \delta})^2 \tilde{G}) \\
&\leq \|\mathbf{g}_t\|^2 - 2\alpha\theta \rho \tilde{G} \\
&\leq (1 - 2\alpha\theta \rho (1 - \sqrt{1 - \delta})^2) \|\mathbf{g}_t\|^2
\end{aligned}$$

With the condition

$$\theta(1 - \rho) \geq \sqrt{1 - \delta} (1 + \sqrt{1 - \delta}) \frac{\tau}{\phi} \left(\frac{1 - \sqrt{2 - \delta}}{1 - \sqrt{1 - \delta}} \right) + \frac{\alpha L}{2\phi^2} (1 + \sqrt{1 - \delta})^2$$

We have value of α

$$\alpha \leq \frac{2\phi^2}{\alpha L(1 + \sqrt{1 - \delta})^2} (\theta(1 - \rho) - \sqrt{1 - \delta}(1 + \sqrt{1 - \delta}) \frac{\tau}{\phi} (\frac{1 - \sqrt{2 - \delta}}{1 - \sqrt{1 - \delta}}))$$

□

Remark 5.7. *Similar to the situation of one round compression, we do not encounter this case in simulation (Section 6.5). The study and analysis of this case is mainly of theoretical importance.*

Case 3 Note that, since we consider compressed gradient along with compression local Hessian gradient product, acquiring the required theoretical guarantee seems quite challenging. Hence we fall back to the one round compression scenario. The local gradients are communicated without any compression and they follow the update rule of Case 3 of one round compression. The convergence result of Theorem 5.7 holds here too.

Remark 5.8. *In Section 6.5, we implement this setting in experiments and observe that this case never happens. Hence, case 3 is not a practical deterrent to the convergence of Algorithm 3 with two round compression.*

5.4 Experimental Result

In this section we provide experimental validation of Algorithm 3. For compression we choose the following scheme: for any given vector $\mathbf{x} \in \mathbb{R}^d$ the compressor outputs $\mathcal{Q}(\mathbf{x}) = \frac{\|\mathbf{x}\|_1}{d} \text{sign}(\mathbf{x})$ where $\text{sign}(\mathbf{x})$ is the quantized vector and $\frac{\|\mathbf{x}\|_1}{d}$ is the scaling factor.

We consider regularized logistic regression for binary classification defined as

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{n} \log(1 + \exp(-y_i \mathbf{x}_i^T \mathbf{w})) + \frac{1}{2n} \|\mathbf{w}\|^2 \quad (5.22)$$

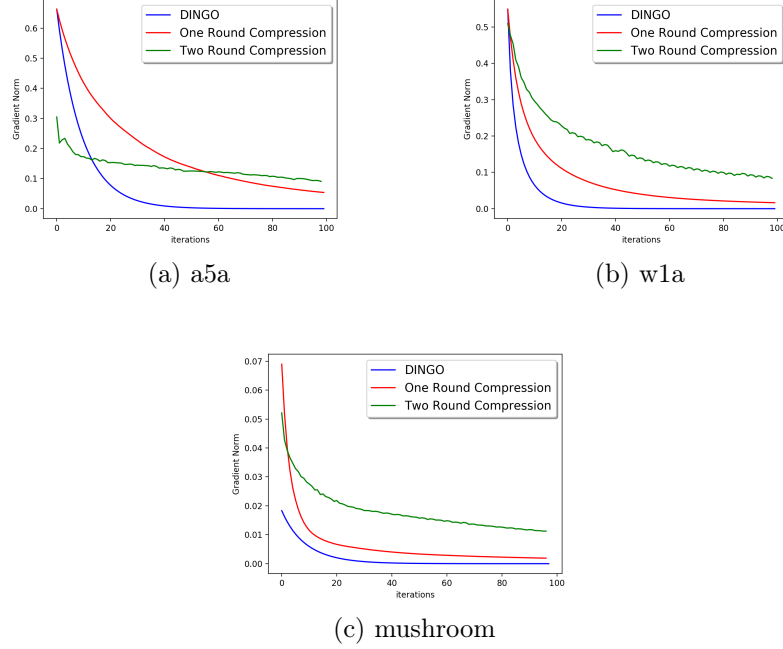


Figure 5.1: Comparison of the convergence of DINGO and one and two round compression methods in terms of $\|\mathbf{g}_t\|$ (gradient norm) of regularized logistic regression for binary classification data.

where $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^d$ are data and $\{y_i\}_{i=1}^n \in \{-1, +1\}$ are the corresponding labels. We choose *a5a* (number of training data, $n = 6414$, dimension $d = 123$), *mushroom* (training data, $n = 8124$, dimension $d = 112$) and *w1a* (training data $n = 2477$, dimension $d = 300$), binary classification datasets from UCI repository [22]. We simulate the distributed set up by partitioning the data into 10 different worker machines. In Figure 5.1, we plot the norm of gradient ($\|\mathbf{g}_t\|$) to validate the optimization method described in the theoretical analysis. We choose $\theta = 0.01$ as defined in Algorithm 3. With this choice of the hyper-parameter, we find that for all the algorithms only case 1 occurs in all the iterations.

Figure 5.1 shows that even with compression, Algorithm 3 converges. We observe that with a decrease in communication cost, the convergence gets slower. DINGO, which has no compression ($\delta = 1$) shows the fastest convergence and two round

compression (with least communication cost) shows the slowest rate. One round compression is in between them in terms of communication and convergence rate.

We also compare the performance of Algorithm 3 (two round compression) with the compressed first order algorithm of [53] with identical learning rates. A threshold of 0.02 on the gradient norm is set as stopping criterion. For the mushroom data [22], we observe that Algorithm 3 communicates a total of 576 bits per machine, whereas [53, Algorithm 1] requires 1008 bits. Hence, a total savings of $432 \times m$ (m : number of machines) or a savings of 43% in bits is achieved. However, for Algorithm 3, the computation complexity at local machines are more than that of first order algorithm. This can be seen as a complexity communication trade-off.

5.5 Conclusion and Future Direction

In this chapter, we address the problem of communication efficiency in distributed second order optimization methods. We consider two different setups with compression in one and two round of the update. A more interesting and challenging idea would be to do one round of communication per iteration and applying compression on the update. An obvious idea is to just use the local gradient and Hessian to compute the update. For any iteration t , in the i -th worker machine, the local gradient and Hessian are denoted by $\mathbf{g}_{i,t}$ and $\mathbf{H}_{i,t}$ respectively. All the worker machines can solve the problem

$$\arg \min_{\hat{\mathbf{p}}_{i,t}} \|\tilde{\mathbf{H}}_{i,t} \hat{\mathbf{p}}_{i,t} + \tilde{\mathbf{g}}_{i,t}\| \quad (5.23)$$

$$\text{such that } \langle \hat{\mathbf{p}}_{i,t}, \mathbf{H}_{i,t} \mathbf{g}_{i,t} \rangle \leq -\theta \|\mathbf{g}_{i,t}\|^2 \quad (5.24)$$

For the purpose analysis, we would need a bound on $\mathbf{H}_{i,t} \mathbf{g}_{i,t}$ as it is used in the worker machine instead of $\mathbf{H}_t \mathbf{g}_t$ (global Hessian and gradient product). Such bound is too strong. In general, Hessian and gradient dissimilarity bound is used separately but

the bound on product is not used. A study on such condition for convergence analysis is an interesting idea. Furthermore, Byzantine resilience on top of compression (later we study in Chapter 6 under more relaxed condition) in the non-convex setting would be a good extension of this work.

CHAPTER 6

DISTRIBUTED NEWTON CAN COMMUNICATE LESS AND RESIST BYZANTINE WORKERS

In this chapter, we propose COMRADE, a distributed approximate Newton-type algorithm that communicates less and is resilient to Byzantine workers. Specifically, we consider a distributed setup with m worker machines and one center machine. The goal is to minimize a regularized convex loss $f : \mathbb{R}^d \rightarrow \mathbb{R}$, which is additive over the available data points. Furthermore, we assume that α fraction of the worker machines are Byzantine, where $\alpha \in [0, 1/2)$. We assume that Byzantine workers can send any arbitrary values to the center machine. In addition, they may completely know the learning algorithm and are allowed to collude with each other.

In our proposed algorithm, the worker machines communicate *only once* per iteration with the center machine. This is in sharp contrast with the state-of-the-art distributed second order algorithms (like GIANT [127], DINGO [32], Determinantal Averaging [37]), which sequentially estimates functions of local gradients and Hessians and communicate them with the center machine. In this way, they end up communicating twice per iteration with the center machine. We show that this sequential estimation is redundant. Instead, in COMRADE, the worker machines only send a d dimensional vector, the product of the inverse of local Hessian and the local gradient. Via sketching arguments, we show that the empirical mean of the product of local Hessian inverse and local gradient is close to the global Hessian inverse and gradient product, and thus just sending the above-mentioned product is sufficient to ensure convergence. Hence, in this way, we save $\mathcal{O}(d)$ bits of communication per iteration. Furthermore, in Section 6.4, we argue that, in order to cut down further

communication, the worker machines can even compress the local Hessian inverse and gradient product. Specifically, we use a (generic) δ -approximate compressor ([71]) for this, and for Byzantine resilience, COMRADE employs a simple thresholding policy on the norms of the local Hessian inverse and local gradient product. Since the norm of the Hessian-inverse and gradient product determines the *amount* of movement for Newton-type algorithms, this norm corresponds to a natural metric for identifying and filtering out Byzantine workers.

Our Contributions: We propose a communication efficient Newton-type algorithm that is robust to Byzantine worker machines. Our proposed algorithm, COMRADE takes as input the local Hessian inverse and gradient product (or a compressed version of it) from the worker machines, and performs a simple thresholding operation on the norm of the said vector to discard $\beta > \alpha$ fraction of workers having largest norm values. We prove the linear-quadratic rate of convergence of our proposed algorithm for strongly convex loss functions. In particular, suppose there are m worker machines, each containing s data points; and let $\Delta_t = \mathbf{w}_t - \mathbf{w}^*$, where \mathbf{w}_t is the t -th iterate of COMRADE, and \mathbf{w}^* is the optimal model we want to estimate. In Theorem 2, we show that

$$\|\Delta_{t+1}\| \leq \max\{\Psi_t^{(1)}\|\Delta_t\|, \Psi_t^{(2)}\|\Delta_t\|^2\} + (\Psi_t^{(3)} + \alpha)\sqrt{\frac{1}{s}},$$

where $\{\Psi_t^{(i)}\}_{i=1}^3$ are quantities dependent on several problem parameters. Notice that the above implies a quadratic rate of convergence when $\|\Delta_t\| \geq \Psi_t^{(1)}/\Psi_t^{(2)}$. Subsequently, when $\|\Delta_t\|$ becomes sufficiently small, the above condition is violated and the convergence slows down to a linear rate. The error-floor, which is $\mathcal{O}(1/\sqrt{s})$ comes from the Byzantine resilience subroutine in conjunction with the simultaneous estimation of Hessian and gradient. Furthermore, in Section 6.4, we consider worker machines compressing the local Hessian inverse and gradient product via a δ -approximate com-

pressor [71], and show that the (order-wise) rate of convergence remain unchanged, and the compression factor, δ affects the constants only.

We experimentally validate our proposed algorithm, COMRADE, with several benchmark data-sets. We consider several types of Byzantine attacks and observe that COMRADE is robust against Byzantine worker machines, yielding better classification accuracy compared to the existing state-of-the-art second order algorithms.

A major technical challenge of this paper is to approximate local gradient and Hessian simultaneously in the presence of Byzantine workers. We use sketching, similar to [127], along with the norm based Byzantine resilience technique. Using *incoherence* (defined shortly) of the local Hessian along with concentration results originating from uniform sampling, we obtain the simultaneous gradient and Hessian approximation. Furthermore, ensuring at least one non-Byzantine machine gets trimmed at every iteration of COMRADE, we control the influence of Byzantine workers.

Related Work: *Second order Optimization:* Second order optimization has received a lot of attention in the recent years in the distributed setting owing to its attractive convergence speed. The fundamentals of second order optimization is laid out in [109], and an extension with better convergence rates is presented in [99]. Recently, in GIANT [127] algorithm, each worker machine computes an approximate Newton direction in each iteration and the center machine averages them to obtain a *globally improved* approximate Newton direction. Furthermore, DINGO [32] generalizes second order optimization beyond convex functions by extending the Newton-MR [102] algorithm in a distributed setting. Very recently, [37] proposes Determinantal averaging to correct the inversion bias of the second order optimization. A slightly different line of work ([126], [57], [95]) uses Hessian sketching to solve a large-scale distributed learning problems.

Byzantine Robust Optimization: In the seminal work of [?], a generic framework of one shot median based robust learning has been proposed and analyzed in the

distributed setting. The issue of Byzantine failure is tackled by grouping the servers in batches and computing the median of batched servers in [27] (the median of means algorithm). Later in [135, 136], co-ordinate wise median, trimmed mean and iterative filtering based algorithm have been proposed and optimal statistical error rate is obtained. Also, [92, 33] consider adversaries may steer convergence to bad local minimizers for non-convex optimization problems. Byzantine resilience with gradient quantization has been addressed in the recent works of [13, 54].

Organization: In Section 6.2, we first analyze COMRADE with *one round* of communication per iteration. We assume $\alpha = 0$, and focus on the communication efficiency aspect only. Subsequently, in Section 7.4, we make $\alpha \neq 0$, thereby addressing communication efficiency and Byzantine resilience simultaneously. Further, in Section 6.4 we augment a compression scheme along with the setting of Section 7.4. Finally, in Section 6.5, we validate our theoretical findings with experiments.

Notation: For a matrix X , we denote $\|X\|_2$ denotes the operator norm, $\sigma_{\max}(X)$ and $\sigma_{\min}(X)$ denote the maximum and minimum singular value. Throughout the paper, we use C, C_1, c, c_1 to denote positive universal constants, whose value changes with instances.

6.1 Problem Formulation

We begin with the standard statistical learning framework for empirical risk minimization, where the objective is to minimize the following loss function:

$$f(\mathbf{w}) = \frac{1}{n} \sum_{j=1}^n \ell_j(\mathbf{w}^T \mathbf{x}_j) + \frac{\lambda}{2} \|\mathbf{w}\|^2, \quad (6.1)$$

where, the loss functions $\ell_j : \mathbb{R} \rightarrow \mathbb{R}, j \in [n]$ are *convex, twice differentiable and smooth*. Moreover, $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n \in \mathbb{R}^d$ denote the input feature vectors and $y_1, y_2, \dots, y_n \in \mathbb{R}$

denote the corresponding responses. Furthermore, we assume that the function f is *strongly convex*, implying the existence of a unique minimizer of (6.1). We denote this minimizer by \mathbf{w}^* . Note that the response $\{y_j\}_{j=1}^n$ is captured by the corresponding loss function $\{\ell_j\}_{j=1}^n$. Some examples of ℓ_j are

$$\text{logistic loss: } \ell_j(z_j) = \log(1 - \exp(-z_j y_j)), \quad \text{squared loss: } \ell_j(z_j) = \frac{1}{2}(z_j - y_j)^2$$

We consider the framework of distributed optimization with m worker machines, where the feature vectors and the loss functions $(\mathbf{x}_1, \ell_1), \dots, (\mathbf{x}_n, \ell_n)$ are partitioned homogeneously among them. Furthermore, we assume that α fraction of the worker machines are Byzantine for some $\alpha < \frac{1}{2}$. The Byzantine machines, by nature, may send any arbitrary values to the center machine. Moreover, they can even collude with each other and plan malicious attacks with complete information of the learning algorithm.

6.2 COMRADE Can Communicate Less

We first present the Newton-type learning algorithm, namely COMRADE without any Byzantine workers, i.e., $\alpha = 0$. It is formally given in Algorithm 4 (with $\beta = 0$). In each iteration of our algorithm, every worker machine computes the local Hessian and local gradient and sends the local second order update (which is the product of the inverse of the local Hessian and local gradient) to the center machine. The center machine aggregates the updates from the worker machines by averaging them and updates the model parameter \mathbf{w} . Later the center machine broadcast the parameter \mathbf{w} to all the worker machines.

In any iteration t , a standard Newton algorithm requires the computation of exact Hessian (\mathbf{H}_t) and gradient (\mathbf{g}_t) of the loss function which can be written as

$$\mathbf{g}_t = \frac{1}{n} \sum_{i=1}^n \ell'_j(\mathbf{w}_t^\top \mathbf{x}_i) \mathbf{x}_i + \lambda \mathbf{w}_t, \quad \mathbf{H}_t = \frac{1}{n} \sum_{i=1}^n \ell''_j(\mathbf{w}_t^\top \mathbf{x}_i) \mathbf{x}_i \mathbf{x}_i^\top + \lambda \mathbf{I}. \quad (6.2)$$

In a distributed set up, the exact Hessian (\mathbf{H}_t) and gradient (\mathbf{g}_t) can be computed in parallel in the following manner. In each iteration, the center machine ‘broadcasts’ the model parameter \mathbf{w}_t to the worker machines and each worker machine computes its own local gradient and Hessian. Then the center machine can compute the exact gradient and exact Hessian by averaging the the local gradient vectors and local Hessian matrices. But for each worker machine the per iteration communication complexity is $\mathcal{O}(d)$ for the gradient computation and $\mathcal{O}(d^2)$ for the Hessian computation. Using Algorithm 4, we reduce the communication cost to only $\mathcal{O}(d)$ per iteration, which is the same as the first order methods.

Each worker machine possess s samples drawn uniformly from $\{(\mathbf{x}_1, \ell_1), (\mathbf{x}_2, \ell_2), \dots, (\mathbf{x}_n, \ell_n)\}$. By S_i , we denote the indices of the samples held by worker machine i . At any iteration t , the worker machine computes the local Hessian $\mathbf{H}_{i,t}$ and local gradient $\mathbf{g}_{i,t}$ as

$$\mathbf{g}_{i,t} = \frac{1}{s} \sum_{i \in S_i} \ell'_j(\mathbf{w}_t^\top \mathbf{x}_i) \mathbf{x}_i + \lambda \mathbf{w}_t, \quad \mathbf{H}_{i,t} = \frac{1}{s} \sum_{i \in S_i} \ell''_j(\mathbf{w}_t^\top \mathbf{x}_i) \mathbf{x}_i \mathbf{x}_i^\top + \lambda \mathbf{I}. \quad (6.3)$$

It is evident from the uniform sampling that $\mathbb{E}[\mathbf{g}_{i,t}] = \mathbf{g}_t$ and $\mathbb{E}[\mathbf{H}_{i,t}] = \mathbf{H}_t$. The update direction from the worker machine is defined as $\hat{\mathbf{p}}_{i,t} = (\mathbf{H}_{i,t})^{-1} \mathbf{g}_{i,t}$. Each worker machine requires $\mathcal{O}(sd^2)$ operations to compute the Hessian matrix $\mathbf{H}_{i,t}$ and $\mathcal{O}(d^3)$ operations to invert the matrix. In practice, the computational cost can be reduced by employing conjugate gradient method. The center machine computes the parameter update direction $\hat{\mathbf{p}}_t = \frac{1}{m} \sum_{i=1}^m \hat{\mathbf{p}}_{i,t}$.

We show that given large enough sample in each worker machine (s is large) and with incoherent data points (the information is spread out and not concentrated to

Algorithm 4 COMMunication-efficient and Robust Approximate Distributed nEWton (COMRADE)

- 1: **Input:** Step size γ , parameter $\beta \geq 0$
 - 2: **Initialize:** Initial iterate $w_0 \in \mathbb{R}^d$
 - 3: **for** $t = 0, 1, \dots, T - 1$ **do**
 - 4: Central machine: broadcasts w_t
 for $i \in [m]$ **do in parallel**
 - 5: i -th worker machine:
 - Non-Byzantine: Computes local gradient $\mathbf{g}_{i,t}$ and local Hessian $\mathbf{H}_{i,t}$; sends $\hat{\mathbf{p}}_{i,t} = (\mathbf{H}_{i,t})^{-1}\mathbf{g}_{i,t}$ to the central machine,
 - Byzantine: Generates \star (arbitrary), and sends it to the center machine
 - end for**
 - 6: Center Machine:
 - Sort the worker machines in a non decreasing order according to norm of updates $\{\hat{\mathbf{p}}_{i,t}\}_{i=1}^m$ from the local machines
 - Return the indices of the first $1 - \beta$ fraction of machines as \mathcal{U}_t ,
 - Approximate Newton Update direction : $\hat{\mathbf{p}}_t = \frac{1}{|\mathcal{U}_t|} \sum_{i \in \mathcal{U}_t} \hat{\mathbf{p}}_{i,t}$
 - Update model parameter: $w_{t+1} = w_t - \gamma \hat{\mathbf{p}}_t$.
 - 7: **end for**
-

a small number of sample data points), the local Hessian $\mathbf{H}_{i,t}$ is close to the global Hessian \mathbf{H}_t in spectral norm, and the local gradient $\mathbf{g}_{i,t}$ is close to the global gradient \mathbf{g}_t . Subsequently, we prove that the empirical average of the local updates acts as a good proxy for the global Newton update and achieves good convergence guarantee.

6.2.1 Theoretical Guarantee

Matrix Sketching: Here we briefly discuss the matrix sketching that is broadly used in the context of *randomized linear algebra*. For any matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ the sketched matrix $\mathbf{Z} \in \mathbb{R}^{s \times d}$ is defined as $\mathbf{S}^T \mathbf{A}$ where $\mathbf{S} \in \mathbb{R}^{n \times s}$ is the sketching matrix (typically $s < n$). Based on the scope and basis of the application, the sketched matrix is constructed by taking linear combination of the rows of matrix which is known as *random projection* or by sampling and scaling a subset of the rows of the matrix which

is known as *random sampling*. The sketching is done to get a smaller representation of the original matrix to reduce computational cost.

Here we consider a uniform row sampling scheme. The matrix \mathbf{Z} is formed by sampling and scaling rows of the matrix \mathbf{A} . Each row of the matrix \mathbf{A} is sampled with probability $p = \frac{1}{n}$ and scaled by multiplying with $\frac{1}{\sqrt{sp}}$.

$$\mathbb{P}\left(\mathbf{z}_i = \frac{\mathbf{a}_j}{\sqrt{sp}}\right) = p,$$

where \mathbf{z}_i is the i -th row matrix \mathbf{Z} and \mathbf{a}_j is the j th row of the matrix \mathbf{A} . Consequently the sketching matrix \mathbf{S} has one non-zero entry in each column.

We define the matrix $\mathbf{A}_t^\top = [\mathbf{a}_1^\top, \dots, \mathbf{a}_n^\top] \in \mathbb{R}^{d \times n}$ where $\mathbf{a}_j = \sqrt{\ell_j''(\mathbf{w}^\top \mathbf{x}_j)} \mathbf{x}_j$. So the exact Hessian in equation (6.2) is $\mathbf{H}_t = \frac{1}{n} \mathbf{A}_t^\top \mathbf{A}_t + \lambda \mathbf{I}$. Assume that S_i is the set of features that are held by the i th worker machine. So the local Hessian is

$$\mathbf{H}_{i,t} = \frac{1}{s} \sum_{j \in S_i} \ell_j''(\mathbf{w}^\top \mathbf{x}_j) \mathbf{x}_j \mathbf{x}_j^\top + \lambda \mathbf{I} = \frac{1}{s} \mathbf{A}_{i,t}^\top \mathbf{A}_{i,t} + \lambda \mathbf{I},$$

where $\mathbf{A}_{i,t} \in \mathbb{R}^{s \times d}$ and the row of the matrix $\mathbf{A}_{i,t}$ is indexed by S_i . Also we define $\mathbf{B}_t = [\mathbf{b}_1, \dots, \mathbf{b}_n] \in \mathbb{R}^{d \times n}$ where $\mathbf{b}_i = \ell_i'(\mathbf{w}^\top \mathbf{x}_i) \mathbf{x}_i$. So the exact gradient in equation (6.2) is $\mathbf{g}_t = \frac{1}{n} \mathbf{B}_t \mathbf{1} + \lambda \mathbf{w}_t$ and the local gradient is

$$\mathbf{g}_{i,t} = \frac{1}{s} \sum_{i \in S_i} \ell_j'(\mathbf{w}_t^\top \mathbf{x}_i) \mathbf{x}_i + \lambda \mathbf{w}_t = \frac{1}{s} \mathbf{B}_{i,t} \mathbf{1} + \lambda \mathbf{w}_t,$$

where $\mathbf{B}_{i,t}$ is the matrix with column indexed by S_i . If $\{\mathbf{S}_i\}_{i=1}^m$ are the sketching matrices then the local Hessian and gradient can be expressed as

$$\mathbf{H}_{i,t} = \mathbf{A}_t^\top \mathbf{S}_i \mathbf{S}_i^\top \mathbf{A}_t + \lambda \mathbf{I} \quad \mathbf{g}_{i,t} = \frac{1}{n} \mathbf{B} \mathbf{S}_i \mathbf{S}_i^\top \mathbf{1} + \lambda \mathbf{w}. \quad (6.4)$$

With the help of sketching idea later we show that the local hessian and gradient are close to the exact hessian and gradient.

The Quadratic function For the purpose of analysis we define an auxiliary quadratic function

$$\phi(\mathbf{p}) = \frac{1}{2}\mathbf{p}^\top \mathbf{H}_t \mathbf{p} - \mathbf{g}_t^\top \mathbf{p} = \frac{1}{2}\mathbf{p}^\top (\mathbf{A}_t^\top \mathbf{A}_t + \lambda \mathbf{I}) \mathbf{p} - \mathbf{g}_t^\top \mathbf{p}. \quad (6.5)$$

The optimal solution to the above function is

$$\mathbf{p}^* = \arg \min \phi(\mathbf{p}) = \mathbf{H}_t^{-1} \mathbf{g}_t = (\mathbf{A}_t^\top \mathbf{A}_t + \lambda \mathbf{I})^{-1} \mathbf{g}_t,$$

which is also the optimal direction of the global Newton update. In this work we consider the local and global (approximate) Newton direction to be

$$\hat{\mathbf{p}}_{i,t} = (\mathbf{A}^\top \mathbf{S}_i \mathbf{S}_i^\top \mathbf{A} + \lambda \mathbf{I})^{-1} \mathbf{g}_{i,t}, \quad \hat{\mathbf{p}}_t = \frac{1}{m} \sum_{i=1}^m \hat{\mathbf{p}}_{i,t}.$$

respectively. And it can be easily verified that each local update $\hat{\mathbf{p}}_{i,t}$ is optimal solution to the following quadratic function

$$\hat{\phi}_{i,t}(p) = \frac{1}{2}\mathbf{p}^\top (\mathbf{A}^\top \mathbf{S}_i \mathbf{S}_i^\top \mathbf{A} + \lambda \mathbf{I}) \mathbf{p} - \mathbf{g}_i^\top \mathbf{p}. \quad (6.6)$$

In our convergence analysis we show that value of the quadratic function in (6.5) with value $\hat{\mathbf{p}}_t$ is close to the optimal value.

Singular Value Decomposition (SVD) For any matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ with rank r , the singular value decomposition is defined as $\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^\top$ where \mathbf{U}, \mathbf{V} are $n \times r$ and $d \times r$ column orthogonal matrices respectively and $\mathbf{\Sigma}$ is a $r \times r$ diagonal matrix with diagonal entries $\{\sigma_1, \dots, \sigma_r\}$. If \mathbf{A} is a symmetric positive semi-definite matrix then $\mathbf{U} = \mathbf{V}$.

We define the matrix $\mathbf{A}_t^\top = [\mathbf{a}_1^\top, \dots, \mathbf{a}_n^\top] \in \mathbb{R}^{d \times n}$ where $\mathbf{a}_j = \sqrt{\ell_j''(\mathbf{w}^\top \mathbf{x}_j)} \mathbf{x}_j$. So the exact Hessian in equation (6.2) is $\mathbf{H}_t = \frac{1}{n} \mathbf{A}_t^\top \mathbf{A}_t + \lambda \mathbf{I}$. Also we define $\mathbf{B}_t =$

$[\mathbf{b}_1, \dots, \mathbf{b}_n] \in \mathbb{R}^{d \times n}$ where $\mathbf{b}_i = \ell'_i(\mathbf{w}^T \mathbf{x}_i) \mathbf{x}_i$. So the exact gradient in equation (6.2) is $\mathbf{g}_t = \frac{1}{n} \mathbf{B}_t \mathbf{1} + \lambda \mathbf{w}_t$

Definition 6.1 (Coherence of a Matrix). *Let $\mathbf{A} \in \mathbb{R}^{n \times d}$ be any matrix with $\mathbf{u} \in \mathbb{R}^{n \times d}$ being its orthonormal basis (the left singular vectors). The row coherence of the matrix \mathbf{A} is defined as $\mu(\mathbf{A}) = \frac{n}{d} \max_i \|\mathbf{u}_i\|^2 \in [1, \frac{n}{d}]$, where \mathbf{u}_i is the i th row of \mathbf{u} .*

Remark 6.1. *If the coherence of \mathbf{A}_t is small, it can be shown that the Hessian matrix can be approximated well via selecting a subset of rows. Note that this is a fairly common to use coherence condition as an approximation tool (see [38, 39, 88])*

In the following, we assume that the Hessian matrix is L -Lipschitz (see definition below), which is a standard assumption for the analysis of the second order method for general smooth loss function (as seen in [127],[37]).

Assumption 6.1. *The Hessian matrix of the loss function f is L -Lipschitz continuous i.e. $\|\nabla^2 f(\mathbf{w}) - \nabla^2 f(\mathbf{w}')\|_2 \leq L \|\mathbf{w} - \mathbf{w}'\|$.*

In the following theorem, we provide the convergence rate of COMRADE (with $\alpha = \beta = 0$) in the terms of $\Delta_t = \mathbf{w}_t - \mathbf{w}^*$. Also, we define $\kappa_t = \sigma_{\max}(\mathbf{H}_t)/\sigma_{\min}(\mathbf{H}_t)$ as the condition number of \mathbf{H}_t , and hence $\kappa_t \geq 1$.

Theorem 6.1. *Let $\mu \in [1, \frac{n}{d}]$ be the coherence of \mathbf{A}_t . Suppose $\gamma = 1$ and $s \geq \frac{3\mu d}{\eta^2} \log \frac{md}{\rho}$ for some $\eta, \rho \in (0, 1)$. Under Assumption 6.1, with probability exceeding $1 - \rho$, we obtain*

$$\|\Delta_{t+1}\| \leq \max\left\{\sqrt{\kappa_t \left(\frac{\zeta^2}{1 - \zeta^2}\right)} \|\Delta_t\|, \frac{L}{\sigma_{\min}(\mathbf{H}_t)} \|\Delta_t\|^2\right\} + \frac{2\epsilon}{\sqrt{\sigma_{\min}(\mathbf{H}_t)}},$$

where $\zeta = \nu(\frac{\eta}{\sqrt{m}} + \frac{\eta^2}{1-\eta})$, $\nu = \frac{\sigma_{\max}(\mathbf{A}^\top \mathbf{A})}{\sigma_{\max}(\mathbf{A}^\top \mathbf{A}) + n\lambda} \leq 1$, and

$$\epsilon = \frac{1}{1 - \eta} \frac{1}{\sqrt{\sigma_{\min}(\mathbf{H}_t)}} \left(1 + \sqrt{2 \ln\left(\frac{m}{\rho}\right)}\right) \sqrt{\frac{1}{s}} \max_i \|\mathbf{b}_i\|. \quad (6.7)$$

Lemma 6.2 (McDiarmid's Inequality). *Let $X = X_1, \dots, X_m$ be m independent random variables taking values from some set A , and assume that $f : A^m \rightarrow \mathbb{R}$ satisfies the following condition (bounded differences):*

$$\sup_{x_1, \dots, x_m, \hat{x}_i} |f(x_1, \dots, x_i, \dots, x_m) - f(x_1, \dots, \hat{x}_i, \dots, x_m)| \leq c_i,$$

for all $i \in \{1, \dots, m\}$. Then for any $\epsilon > 0$ we have

$$P[f(X_1, \dots, X_m) - \mathbb{E}[f(X_1, \dots, X_m)] \geq \epsilon] \leq \exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^m c_i^2}\right).$$

The property described in the following Lemma 6.3 is a very useful result for uniform row sampling sketching matrix.

Lemma 6.3 (Lemma 8 [127]). *Let $\eta, \delta \in (0, 1)$ be a fixed parameter and $r = \text{rank}(\mathbf{A}_t)$ and $\mathbf{U} \in \mathbb{R}^{n \times r}$ be the orthonormal bases of the matrix \mathbf{A}_t . Let $\{\mathbf{S}_i\}_{i=1}^m$ be sketching matrices and $\mathbf{S} = \frac{1}{\sqrt{m}}[\mathbf{S}_1, \dots, \mathbf{S}_m] \in \mathbb{R}^{n \times ms}$. With probability $1 - \delta$ the following holds*

$$\|\mathbf{U}^\top \mathbf{S}_i \mathbf{S}_i^\top \mathbf{U} - \mathbf{I}\|_2 \leq \eta \quad \forall i \in [m] \quad \text{and} \quad \|\mathbf{U}^\top \mathbf{S} \mathbf{S}^\top \mathbf{U} - \mathbf{I}\|_2 \leq \frac{\eta}{\sqrt{m}}.$$

Lemma 6.4. *Let $\mathbf{S} \in \mathbb{R}^{n \times s}$ be any uniform sampling sketching matrix, then for any matrix $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_n] \in \mathbb{R}^{d \times n}$ with probability $1 - \delta$ for any $\delta > 0$ we have,*

$$\left\| \frac{1}{n} \mathbf{B} \mathbf{S} \mathbf{S}^\top \mathbf{1} - \frac{1}{n} \mathbf{B} \mathbf{1} \right\| \leq \left(1 + \sqrt{2 \ln\left(\frac{1}{\delta}\right)}\right) \sqrt{\frac{1}{s}} \max_i \|\mathbf{b}_i\|,$$

where $\mathbf{1}$ is all ones vector.

Proof. The vector $\mathbf{B} \mathbf{1}$ is the sum of column of the matrix \mathbf{B} and $\mathbf{B} \mathbf{S} \mathbf{S}^\top \mathbf{1}$ is the sum of uniformly sampled and scaled column of the matrix \mathbf{B} where the scaling factor is $\frac{1}{\sqrt{sp}}$ with $p = \frac{1}{n}$. If (i_1, \dots, i_s) is the set of sampled indices then $\mathbf{B} \mathbf{S} \mathbf{S}^\top \mathbf{1} = \sum_{k \in (i_1, \dots, i_s)} \frac{1}{sp} \mathbf{b}_k$.

Define the function $f(i_1, \dots, i_s) = \|\frac{1}{n}\mathbf{BSS}^\top \mathbf{1} - \frac{1}{n}\mathbf{B}\mathbf{1}\|$. Now consider a sampled set $(i_1, \dots, i_{j'}, \dots, i_s)$ with only one item (column) replaced then the bounded difference is

$$\begin{aligned}\Delta &= |f(i_1, \dots, i_j, \dots, i_s) - f(i_1, \dots, i_{j'}, \dots, i_s)| \\ &= \left| \frac{1}{n} \left\| \frac{1}{sp} \mathbf{b}_{i_{j'}} - \frac{1}{sp} \mathbf{b}_{i_j} \right\| \right| \leq \frac{2}{s} \max_i \|\mathbf{b}_i\|.\end{aligned}$$

Now we have the expectation

$$\begin{aligned}\mathbb{E}[\|\frac{1}{n}\mathbf{BSS}^\top \mathbf{1} - \frac{1}{n}\mathbf{B}\mathbf{1}\|^2] &\leq \frac{n}{sn^2} \sum_{i=1}^n \|\mathbf{b}_i\|^2 = \frac{1}{s} \max_i \|\mathbf{b}_i\|^2 \\ \Rightarrow \mathbb{E}[\|\frac{1}{n}\mathbf{BSS}^\top \mathbf{1} - \frac{1}{n}\mathbf{B}\mathbf{1}\|] &\leq \sqrt{\frac{1}{s} \max_i \|\mathbf{b}_i\|}.\end{aligned}$$

Using McDiarmid inequality (Lemma 6.2) we have

$$P\left[\left\|\frac{1}{n}\mathbf{BSS}^\top \mathbf{1} - \frac{1}{n}\mathbf{B}\mathbf{1}\right\| \geq \sqrt{\frac{1}{s} \max_i \|\mathbf{b}_i\|} + t\right] \leq \exp\left(-\frac{2t^2}{s\Delta^2}\right).$$

Equating the probability with δ we have

$$\begin{aligned}\exp\left(-\frac{2t^2}{s\Delta^2}\right) &= \delta \\ \Rightarrow t &= \Delta \sqrt{\frac{s}{2} \ln\left(\frac{1}{\delta}\right)} = \max_i \|\mathbf{b}_i\| \sqrt{\frac{2}{s} \ln\left(\frac{1}{\delta}\right)}.\end{aligned}$$

Finally we have with probability $1 - \delta$

$$\left\|\frac{1}{n}\mathbf{BSS}^\top \mathbf{1} - \frac{1}{n}\mathbf{B}\mathbf{1}\right\| \leq \left(1 + \sqrt{2 \ln\left(\frac{1}{\delta}\right)}\right) \sqrt{\frac{1}{s} \max_i \|\mathbf{b}_i\|}.$$

□

Remark 6.2. For m sketching matrix $\{\mathbf{S}_i\}_{i=1}^m$, the bound in the Lemma 6.4 is

$$\left\| \frac{1}{n} \mathbf{B} \mathbf{S}_i \mathbf{S}_i^\top \mathbf{1} - \frac{1}{n} \mathbf{B} \mathbf{1} \right\| \leq (1 + \sqrt{2 \ln(\frac{m}{\delta})}) \sqrt{\frac{1}{s}} \max_i \|\mathbf{b}_i\|,$$

with probability $1 - \delta$ for any $\delta > 0$ for all $i \in \{1, 2, \dots, m\}$. In the case that each worker machine holds data based on the uniform sketching matrix the local gradient is close to the exact gradient. Thus the local second order update acts as a good approximate to the exact Newton update.

Now we consider the update rule of GIANT [127] where the update is done in two rounds in each iteration. In the first round each worker machine computes and send the local gradient and the center machine computes the exact gradient \mathbf{g}_t in iteration t . Next the center machine broadcasts the exact gradient and each worker machine computes the local Hessian and send $\tilde{\mathbf{p}}_{i,t} = (\mathbf{H}_{i,t})^{-1} \mathbf{g}_t$ to the center machine and the center machine computes the approximate Newton direction $\tilde{\mathbf{p}}_t = \frac{1}{m} \sum_{i=1}^m \tilde{\mathbf{p}}_{i,t}$. Now based on this we restate the following lemma (Lemma 6 [127]).

Lemma 6.5. Let $\{\mathbf{S}_i\}_{i=1}^m \in \mathbb{R}^{n \times s}$ be sketching matrices based on Lemma 6.3. Let ϕ_t be defined in (6.5) and $\tilde{\mathbf{p}}_t$ be the update. It holds that

$$\min_{\mathbf{p}} \phi_t(\mathbf{p}) \leq \phi_t(\tilde{\mathbf{p}}_t) \leq (1 - \zeta^2) \min_{\mathbf{p}} \phi_t(\mathbf{p}),$$

where $\zeta = \nu(\frac{\eta}{\sqrt{m}} + \frac{\eta^2}{1-\eta})$ and $\nu = \frac{\sigma_{\max}(\mathbf{A}^\top \mathbf{A})}{\sigma_{\max}(\mathbf{A}^\top \mathbf{A}) + n\lambda} \leq 1$.

Now we prove similar guarantee for the update according to COMRADE in Algorithm 4.

Lemma 6.6. Let $\{\mathbf{S}_i\}_{i=1}^m \in \mathbb{R}^{n \times s}$ be sketching matrices based on Lemma 6.3. Let ϕ_t be defined in (6.5) and $\hat{\mathbf{p}}_t$ be defined in Algorithm 4 ($\beta = 0$)

$$\min_{\mathbf{p}} \phi_t(\mathbf{p}) \leq \phi_t(\hat{\mathbf{p}}_t) \leq \epsilon^2 + (1 - \zeta^2) \min_{\mathbf{p}} \phi_t(\mathbf{p}),$$

where $\epsilon = \frac{1}{1-\eta} \frac{1}{\sqrt{\sigma_{\min}(\mathbf{H}_t)}} (1 + \sqrt{2 \ln(\frac{m}{\delta})}) \sqrt{\frac{1}{s}} \max_i \|\mathbf{b}_i\|$ and $\zeta = \nu(\frac{\eta}{\sqrt{m}} + \frac{\eta^2}{1-\eta})$ and $\nu = \frac{\sigma_{\max}(\mathbf{A}^\top \mathbf{A})}{\sigma_{\max}(\mathbf{A}^\top \mathbf{A}) + n\lambda}$.

Proof. First consider the quadratic function (6.5)

$$\begin{aligned} \phi_t(\hat{\mathbf{p}}_t) - \phi_t(\mathbf{p}^*) &= \frac{1}{2} \|\mathbf{H}_t^{\frac{1}{2}}(\hat{\mathbf{p}}_t - \mathbf{p}^*)\|^2 \\ &\leq \underbrace{(\|\mathbf{H}_t^{\frac{1}{2}}(\hat{\mathbf{p}}_t - \tilde{\mathbf{p}}_t)\|^2)}_{\text{Term1}} + \underbrace{(\|\mathbf{H}_t^{\frac{1}{2}}(\tilde{\mathbf{p}}_t - \mathbf{p}^*)\|^2)}_{\text{Term2}}, \end{aligned} \quad (6.8)$$

where $\tilde{\mathbf{p}}_t = \frac{1}{m} \sum_{i=1}^m (\mathbf{H}_{i,t})^{-1} \mathbf{g}_t$. First we bound the Term 2 of (6.8) using the quadratic function and Lemma 6.5

$$\begin{aligned} \frac{1}{2} \left\| \mathbf{H}_t^{\frac{1}{2}}(\tilde{\mathbf{p}}_t - \mathbf{p}^*) \right\|^2 &\leq \zeta^2 \left\| \mathbf{H}_t^{\frac{1}{2}} \mathbf{p}^* \right\|^2 \quad (\text{Using Lemma 6.5}) \\ &= -\zeta^2 \phi_t(\mathbf{p}^*). \end{aligned} \quad (6.9)$$

The step in equation (6.9) is from the definition of the function ϕ_t and \mathbf{p}^* . It can be shown that

$$\phi_t(\mathbf{p}^*) = - \left\| \mathbf{H}_t^{\frac{1}{2}} \mathbf{p}^* \right\|^2.$$

Now we bound the Term 1 in (6.8). By Lemma 6.3, we have $(1 - \eta) \mathbf{A}_t^\top \mathbf{A}_t \preceq \mathbf{A}_t^\top \mathbf{S}_i \mathbf{S}_i^\top \mathbf{A}_t \preceq (1 + \eta) \mathbf{A}_t^\top \mathbf{A}_t$. Following we have $(1 - \eta) \mathbf{H}_t \preceq \mathbf{H}_{i,t} \preceq (1 + \eta) \mathbf{H}_t$. Thus there exists matrix ξ_i satisfying

$$\mathbf{H}_t^{\frac{1}{2}} \mathbf{H}_{i,t}^{-1} \mathbf{H}_t^{\frac{1}{2}} = \mathbf{I} + \xi_i \quad \text{and} \quad -\frac{\eta}{1+\eta} \preceq \xi_i \preceq \frac{\eta}{1-\eta},$$

So we have,

$$\left\| \mathbf{H}_t^{\frac{1}{2}} \mathbf{H}_{i,t}^{-1} \mathbf{H}_t^{\frac{1}{2}} \right\| \leq 1 + \frac{\eta}{1-\eta} = \frac{1}{1-\eta}. \quad (6.10)$$

Now we have

$$\begin{aligned} \left\| \mathbf{H}_t^{\frac{1}{2}} (\hat{\mathbf{p}}_t - \tilde{\mathbf{p}}_t) \right\| &= \left\| \mathbf{H}_t^{\frac{1}{2}} \frac{1}{m} \sum_{i=1}^m (\hat{\mathbf{p}}_{i,t} - \tilde{\mathbf{p}}_{i,t}) \right\| \\ &\leq \frac{1}{m} \sum_{i=1}^m \left\| \mathbf{H}_t^{\frac{1}{2}} (\hat{\mathbf{p}}_{i,t} - \tilde{\mathbf{p}}_{i,t}) \right\| \\ &= \frac{1}{m} \sum_{i=1}^m \left\| \mathbf{H}_t^{\frac{1}{2}} \mathbf{H}_{i,t}^{-1} (\mathbf{g}_{i,t} - \mathbf{g}_t) \right\| \\ &= \frac{1}{m} \sum_{i=1}^m \left\| \mathbf{H}_t^{\frac{1}{2}} \mathbf{H}_{i,t}^{-1} \mathbf{H}_t^{\frac{1}{2}} \mathbf{H}_t^{-\frac{1}{2}} (\mathbf{g}_{i,t} - \mathbf{g}_t) \right\| \\ &\leq \frac{1}{m} \sum_{i=1}^m \left\| \mathbf{H}_t^{\frac{1}{2}} \mathbf{H}_{i,t}^{-1} \mathbf{H}_t^{\frac{1}{2}} \right\| \left\| \mathbf{H}_t^{-\frac{1}{2}} (\mathbf{g}_{i,t} - \mathbf{g}_t) \right\| \\ &\leq \frac{1}{1-\eta} \frac{1}{m} \sum_{i=1}^m \left\| \mathbf{H}_t^{-\frac{1}{2}} (\mathbf{g}_{i,t} - \mathbf{g}_t) \right\| \quad (\text{Using (6.10)}) \\ &\leq \frac{1}{1-\eta} \frac{1}{\sqrt{\sigma_{\min}(\mathbf{H}_t)}} \frac{1}{m} \sum_{i=1}^m \left\| \mathbf{g}_{i,t} - \mathbf{g}_t \right\|. \end{aligned} \quad (6.11)$$

Now we bound $\left\| \mathbf{g}_{i,t} - \mathbf{g}_t \right\|$ using Lemma 6.4,

$$\left\| \mathbf{g}_{i,t} - \mathbf{g}_t \right\| = \left\| \frac{1}{n} \mathbf{B} \mathbf{S} \mathbf{S}^\top \mathbf{1} - \frac{1}{n} \mathbf{B} \mathbf{1} \right\| \leq (1 + \sqrt{2 \ln(\frac{m}{\delta})}) \sqrt{\frac{1}{s}} \max_i \left\| \mathbf{b}_i \right\|.$$

Plugging it into equation (6.11) we get,

$$\begin{aligned} \left\| \mathbf{H}_t^{\frac{1}{2}} (\hat{\mathbf{p}}_t - \tilde{\mathbf{p}}_t) \right\| &\leq \frac{1}{1-\eta} \frac{1}{\sqrt{\sigma_{\min}(\mathbf{H}_t)}} \frac{1}{m} \sum_{i=1}^m \left\| \mathbf{g}_{i,t} - \mathbf{g}_t \right\| \\ &\leq \frac{1}{1-\eta} \frac{1}{\sqrt{\sigma_{\min}(\mathbf{H}_t)}} (1 + \sqrt{2 \ln(\frac{m}{\delta})}) \sqrt{\frac{1}{s}} \max_i \left\| \mathbf{b}_i \right\|. \end{aligned} \quad (6.12)$$

Now collecting the terms of (6.12) and (6.9) and plugging them into (6.8) we have

$$\begin{aligned}\phi_t(\hat{\mathbf{p}}_t) - \phi_t(\mathbf{p}^*) &\leq \epsilon^2 - \zeta^2 \phi_t(\mathbf{p}^*) \\ \Rightarrow \phi_t(\hat{\mathbf{p}}_t) &\leq \epsilon^2 + (1 - \zeta^2) \phi_t(\mathbf{p}^*),\end{aligned}$$

where ϵ is as defined in (6.7). □

Lemma 6.7. *Let $\zeta \in (0, 1)$, ϵ be any fixed parameter. And $\hat{\mathbf{p}}_t$ satisfies $\phi_t(\hat{\mathbf{p}}_t) \leq \epsilon^2 + (1 - \zeta^2) \min_{\mathbf{p}} \phi_t(\mathbf{p})$. Under the Assumption 6.1 (Hessian L -Lipschitz) and $\Delta_t = \mathbf{w}_t - \mathbf{w}^*$ satisfies*

$$\Delta_{t+1}^\top \mathbf{H}_t \Delta_{t+1} \leq L \|\Delta_{t+1}\| \|\Delta_t\|^2 + \frac{\zeta^2}{1 - \zeta^2} \Delta_t^\top \mathbf{H}_t \Delta_t + 2\epsilon^2.$$

Proof. We have $\mathbf{w}_{t+1} = \mathbf{w}_t - \hat{\mathbf{p}}_t$, $\Delta_t = \mathbf{w}_t - \mathbf{w}^*$ and $\Delta_{t+1} = \mathbf{w}_{t+1} - \mathbf{w}^*$. Also $\hat{\mathbf{p}}_t = \mathbf{w}_t - \mathbf{w}_{t+1} = \Delta_t - \Delta_{t+1}$. From the definition of ϕ we have,

$$\begin{aligned}\phi_t(\hat{\mathbf{p}}_t) &= \frac{1}{2} (\Delta_t - \Delta_{t+1})^\top \mathbf{H}_t (\Delta_t - \Delta_{t+1}) - (\Delta_t - \Delta_{t+1})^\top \mathbf{g}_t, \\ (1 - \zeta^2) \phi_t\left(\frac{1}{(1 - \zeta^2)} \Delta_t\right) &= \frac{1}{2(1 - \zeta^2)} \Delta_t^\top \mathbf{H}_t \Delta_t - \Delta_t^\top \mathbf{g}_t.\end{aligned}$$

From the above two equation we have

$$\begin{aligned}\phi_t(\hat{\mathbf{p}}_t) - (1 - \zeta^2) \phi_t\left(\frac{1}{(1 - \zeta^2)} \Delta_t\right) &= \frac{1}{2} \Delta_{t+1}^\top \mathbf{H}_t \Delta_{t+1} - \frac{1}{2} \Delta_t^\top \mathbf{H}_t \Delta_{t+1} + \frac{1}{2} \Delta_{t+1}^\top \mathbf{g}_t - \frac{\zeta^2}{2(1 - \zeta^2)} \Delta_t^\top \mathbf{H}_t \Delta_t.\end{aligned}$$

From Lemma 6.6 the following holds

$$\begin{aligned}\phi_t(\hat{\mathbf{p}}_t) &\leq \epsilon^2 + (1 - \zeta^2) \min_{\mathbf{p}} \phi_t(\mathbf{p}) \\ &\leq \epsilon^2 + (1 - \zeta^2) \phi_t\left(\frac{1}{(1 - \zeta^2)} \Delta_t\right).\end{aligned}$$

So we have

$$\frac{1}{2}\Delta_{t+1}^\top \mathbf{H}_t \Delta_{t+1} - \Delta_t^\top \mathbf{H}_t \Delta_{t+1} + \Delta_{t+1}^\top \mathbf{g}_t - \frac{\zeta^2}{2(1-\zeta^2)} \Delta_t^\top \mathbf{H}_t \Delta_t \leq \epsilon^2. \quad (6.13)$$

Consider $\mathbf{g}_t = \mathbf{g}(\mathbf{w}_t)$

$$\begin{aligned} \mathbf{g}(\mathbf{w}_t) &= \mathbf{g}(\mathbf{w}^*) + \left(\int_0^1 \nabla^2 f(\mathbf{w}^* + z(\mathbf{w}_t - \mathbf{w}^*)) dz \right) (\mathbf{w}_t - \mathbf{w}^*) \\ &= \left(\int_0^1 \nabla^2 f(\mathbf{w}^* + z(\mathbf{w}_t - \mathbf{w}^*)) dz \right) \Delta_t \quad (\text{as } \mathbf{g}(\mathbf{w}^*) = 0). \end{aligned}$$

Now we bound the following

$$\begin{aligned} \|\mathbf{H}_t \Delta_t - \mathbf{g}(\mathbf{w}_t)\| &\leq \|\Delta_t\| \left\| \int_0^1 [\nabla^2 f(\mathbf{w}_t) - \nabla^2 f(\mathbf{w}^* + z(\mathbf{w}_t - \mathbf{w}^*))] dz \right\| \\ &\leq \|\Delta_t\| \int_0^1 \|\nabla^2 f(\mathbf{w}_t) - \nabla^2 f(\mathbf{w}^* + z(\mathbf{w}_t - \mathbf{w}^*))\| dz \quad (\text{By Jensen's Inequality}) \\ &\leq \|\Delta_t\| \int_0^1 (1-z)L \|\mathbf{w}_t - \mathbf{w}^*\| dz \quad (\text{by } L\text{-Lipschitz assumption}) \\ &= \frac{L}{2} \|\Delta_t\|^2. \end{aligned}$$

Plugging it into (6.13) we have

$$\begin{aligned} \Delta_{t+1}^\top \mathbf{H}_t \Delta_{t+1} &\leq 2\Delta_{t+1}^\top (\mathbf{H}_t \Delta_t - \mathbf{g}_t) + \frac{\zeta^2}{(1-\zeta^2)} \Delta_t^\top \mathbf{H}_t \Delta_t + 2\epsilon^2 \\ &\leq 2\|\Delta_{t+1}\| \|\mathbf{H}_t \Delta_t - \mathbf{g}_t\| + \frac{\zeta^2}{(1-\zeta^2)} \Delta_t^\top \mathbf{H}_t \Delta_t + 2\epsilon^2 \\ &\leq L\|\Delta_{t+1}\| \|\Delta_t\|^2 + \frac{\zeta^2}{(1-\zeta^2)} \Delta_t^\top \mathbf{H}_t \Delta_t + 2\epsilon^2. \end{aligned}$$

□

Proof. Proof of Theorem 6.1 From the Lemma 6.7 with probability $1 - \delta$

$$\begin{aligned}\Delta_{t+1}^\top \mathbf{H}_t \Delta_{t+1} &\leq L \|\Delta_{t+1}\| \|\Delta_t\|^2 + \frac{\zeta^2}{(1 - \zeta^2)} \Delta_t^\top \mathbf{H}_t \Delta_t + 2\epsilon^2 \\ &\leq L \|\Delta_{t+1}\| \|\Delta_t\|^2 + \left(\frac{\zeta^2}{1 - \zeta^2} \sigma_{\max}(\mathbf{H}_t)\right) \|\Delta_t\|^2 + 2\epsilon^2.\end{aligned}$$

So we have,

$$\|\Delta_{t+1}\| \leq \max\left\{\sqrt{\frac{\sigma_{\max}(\mathbf{H}_t)}{\sigma_{\min}(\mathbf{H}_t)}} \left(\frac{\zeta^2}{1 - \zeta^2}\right) \|\Delta_t\|, \frac{L}{\sigma_{\min}(\mathbf{H}_t)} \|\Delta_t\|^2\right\} + \frac{2\epsilon}{\sqrt{\sigma_{\min}(\mathbf{H}_t)}}.$$

□

Remark 6.3. *It is well known that a distributed Newton method has linear-quadratic convergence rate. In Theorem 6.1 the quadratic term comes from the standard analysis of Newton method. The linear term (which is small) arises owing to Hessian approximation. It gets smaller with better Hessian approximation (smaller η), and thus the above rate becomes quadratic one. The small error floor arises due to the gradient approximation in the worker machines, which is essential for the one round of communication per iteration. The error floor is $\propto \frac{1}{\sqrt{s}}$ where s is the number of samples in each worker machine. So for a sufficiently large s , the error floor becomes negligible.*

Remark 6.4. *The sample size in each worker machine is dependent on the coherence of the matrix \mathbf{A}_t and the dimension d of the problem. Theoretically, the analysis is feasible for the case of $s \geq d$ (since we work with $\mathbf{H}_{i,t}^{-1}$). However, when $s < d$, one can replace the inverse by a pseudo-inverse (modulo some changes in convergence rate).*

6.3 COMRADE Can Resist Byzantine Workers

In this section, we analyze COMRADE with Byzantine workers. We assume that $\alpha (< 1/2)$ fraction of worker machines are Byzantine. We define the set of Byzantine worker machines by \mathcal{B} and the set of the good (non-Byzantine) machines by \mathcal{M} . COMRADE employs a ‘norm based thresholding’ scheme on the local Hessian inverse and gradient product to tackle the Byzantine workers.

In the t -th iteration, the center machine outputs a set \mathcal{U}_t with $|\mathcal{U}_t| = (1 - \beta)m$, consisting the indices of the worker machines with smallest norm. Hence, we ‘trim’ the worker machines that may try to diverge the learning algorithm. We denote the set of trimmed machines as \mathcal{T}_t . Moreover, we take $\beta > \alpha$ to ensure at least one good machine falls in \mathcal{T}_t . This condition helps us to control the Byzantine worker machines. Finally, the update is given by $\hat{\mathbf{p}}_t = \frac{1}{|\mathcal{U}_t|} \sum_{i \in \mathcal{U}_t} \hat{\mathbf{p}}_{i,t}$. We define:

$$\epsilon_{byz}^2 = [3(\frac{1-\alpha}{1-\beta})^2 + 4\kappa_t(\frac{\alpha}{1-\beta})^2]\epsilon^2, \quad (6.14)$$

$$\zeta_{byz}^2 = 2(\frac{1-\alpha}{1-\beta})^2(\frac{\nu}{1-\eta})^2 + \nu^2(\frac{1-\alpha}{1-\beta})^2(\frac{\eta}{\sqrt{(1-\alpha)m}} + \frac{\eta^2}{1-\eta})^2 + 4\kappa_t(\frac{\alpha}{1-\beta})^2[2 + (\frac{\nu}{1-\eta})^2]. \quad (6.15)$$

ϵ is defined in (6.7), $\nu = \frac{\sigma_{max}(\mathbf{A}^T \mathbf{A})}{\sigma_{max}(\mathbf{A}^T \mathbf{A}) + n\lambda}$ and κ_t is the condition number of the exact Hessian \mathbf{H}_t .

Theorem 6.8. *Let $\mu \in [1, \frac{n}{d}]$ be the coherence of \mathbf{A}_t . Suppose $\gamma = 1$ and $s \geq \frac{3\mu d}{\eta^2} \log \frac{md}{\rho}$ for some $\eta, \rho \in (0, 1)$. For $0 \leq \alpha < \beta < 1/2$, under Assumption 6.1, with probability exceeding $1 - \rho$, Algorithm 4 yields*

$$\|\Delta_{t+1}\| \leq \max\{\sqrt{\kappa_t(\frac{\zeta_{byz}^2}{1 - \zeta_{byz}^2})}\|\Delta_t\|, \frac{L}{\sigma_{min}(\mathbf{H}_t)}\|\Delta_t\|^2\} + \frac{2\epsilon_{byz}}{\sqrt{\sigma_{min}(\mathbf{H}_t)}},$$

where ζ_{byz} and ϵ_{byz} are defined in equations (6.14) and (6.15) respectively.

Lemma 6.9. Let $\{\mathbf{S}_i\}_{i=1}^m \in \mathbb{R}^{n \times s}$ be sketching matrices based on Lemma 6.3. Let ϕ_t be defined in (6.5) and $\hat{\mathbf{p}}_t$ be defined in Algorithm 4. It holds that

$$\min_{\mathbf{p}} \phi_t(\mathbf{p}) \leq \phi_t(\hat{\mathbf{p}}_t) \leq \epsilon_{byz}^2 + (1 - \zeta_{byz}^2) \phi(\mathbf{p}^*),$$

where ϵ_{byz} and ζ_{byz} is defined in (6.14) and (6.15) respectively.

Proof. In the following analysis we omit the subscript 't'. From the definition of the quadratic function (6.5) we know that

$$\phi(\hat{\mathbf{p}}) - \phi(\mathbf{p}^*) = \frac{1}{2} \|\mathbf{H}^{\frac{1}{2}}(\hat{\mathbf{p}} - \mathbf{p}^*)\|^2.$$

Now we consider

$$\begin{aligned} & \frac{1}{2} \|\mathbf{H}^{\frac{1}{2}}(\hat{\mathbf{p}} - \mathbf{p}^*)\|^2 \\ &= \frac{1}{2} \|\mathbf{H}^{\frac{1}{2}}(\frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} \hat{\mathbf{p}}_i - \mathbf{p}^*)\|^2 \\ &= \frac{1}{2} \|\mathbf{H}^{\frac{1}{2}} \frac{1}{|\mathcal{U}|} (\sum_{i \in \mathcal{M}} (\hat{\mathbf{p}}_i - \mathbf{p}^*) - \sum_{i \in (\mathcal{M} \cap \mathcal{T})} (\hat{\mathbf{p}}_i - \mathbf{p}^*) + \sum_{i \in (\mathcal{U} \cap \mathcal{B})} (\hat{\mathbf{p}}_i - \mathbf{p}^*))\|^2 \\ &\leq \underbrace{\|\mathbf{H}^{\frac{1}{2}} \frac{1}{|\mathcal{U}|} (\sum_{i \in \mathcal{M}} (\hat{\mathbf{p}}_i - \mathbf{p}^*))\|^2}_{Term1} + \underbrace{2\|\mathbf{H}^{\frac{1}{2}} \frac{1}{|\mathcal{U}|} \sum_{i \in (\mathcal{M} \cap \mathcal{T})} (\hat{\mathbf{p}}_i - \mathbf{p}^*)\|^2}_{Term2} + \underbrace{2\|\mathbf{H}^{\frac{1}{2}} \frac{1}{|\mathcal{U}|} \sum_{i \in (\mathcal{U} \cap \mathcal{B})} (\hat{\mathbf{p}}_i - \mathbf{p}^*)\|^2}_{Term3}. \end{aligned}$$

Now we bound each term separately and use the result of the Lemma 6.6 to bound each term.

$$\begin{aligned} Term1 &= \|\mathbf{H}^{\frac{1}{2}} \frac{1}{|\mathcal{U}|} (\sum_{i \in \mathcal{M}} (\hat{\mathbf{p}}_i - \mathbf{p}^*))\|^2 \\ &= (\frac{1-\alpha}{1-\beta})^2 \|\mathbf{H}^{\frac{1}{2}} \frac{1}{|\mathcal{M}|} (\sum_{i \in \mathcal{M}} (\hat{\mathbf{p}}_i - \mathbf{p}^*))\|^2 \\ &\leq (\frac{1-\alpha}{1-\beta})^2 [\epsilon^2 + \zeta_{\mathcal{M}}^2 \|\mathbf{H}^{\frac{1}{2}} \mathbf{p}^*\|^2], \end{aligned}$$

where $\zeta_{\mathcal{M}} = \nu(\frac{\eta}{\sqrt{|\mathcal{M}|}} + \frac{\eta^2}{1-\eta}) = \nu(\frac{\eta}{\sqrt{(1-\alpha)m}} + \frac{\eta^2}{1-\eta})$.

Similarly the Term 2 can be bonded as it is a bound on good machines

$$\begin{aligned}
Term2 &= 2\|\mathbf{H}^{\frac{1}{2}}\frac{1}{|\mathcal{U}|}\sum_{i\in(\mathcal{M}\cap\mathcal{T})}(\hat{\mathbf{p}}_i - \mathbf{p}^*)\|^2 \\
&= 2\left(\frac{1-\alpha}{1-\beta}\right)^2\|\mathbf{H}^{\frac{1}{2}}\frac{1}{|\mathcal{M}\cap\mathcal{T}|}\sum_{i\in(\mathcal{M}\cap\mathcal{T})}(\hat{\mathbf{p}}_i - \mathbf{p}^*)\|^2 \\
&\leq 2\left(\frac{1-\alpha}{1-\beta}\right)^2[\epsilon^2 + \zeta_{\mathcal{M}\cap\mathcal{T}}^2\|\mathbf{H}^{\frac{1}{2}}\mathbf{p}^*\|^2],
\end{aligned}$$

where $\zeta_{\mathcal{M}\cap\mathcal{T}} = \nu\left(\frac{\eta}{\sqrt{|\mathcal{M}\cap\mathcal{T}|}} + \frac{\eta^2}{1-\eta}\right) \leq \nu\left(\frac{\eta}{\sqrt{(1-\beta)m}} + \frac{\eta^2}{1-\eta}\right)$.

For the Term 3 we know that $\beta > \alpha$ so all the untrimmed worker norm is bounded by a good machine as at least one good machine gets trimmed.

$$\begin{aligned}
Term3 &= 2\|\mathbf{H}^{\frac{1}{2}}\frac{1}{|\mathcal{U}|}\sum_{i\in(\mathcal{U}\cap\mathcal{B})}(\hat{\mathbf{p}}_i - \mathbf{p}^*)\|^2 \\
&\leq 2\sigma_{max}(\mathbf{H})\left(\frac{|\mathcal{U}\cap\mathcal{B}|}{|\mathcal{U}|}\right)^2\left\|\frac{1}{|\mathcal{U}\cap\mathcal{B}|}\sum_{i\in(\mathcal{U}\cap\mathcal{B})}(\hat{\mathbf{p}}_i - \mathbf{p}^*)\right\|^2 \\
&\leq 2\sigma_{max}(\mathbf{H})\left(\frac{|\mathcal{U}\cap\mathcal{B}|}{|\mathcal{U}|}\right)^2\frac{1}{|\mathcal{U}\cap\mathcal{B}|}\sum_{i\in(\mathcal{U}\cap\mathcal{B})}\|(\hat{\mathbf{p}}_i - \mathbf{p}^*)\|^2 \\
&\leq 4\sigma_{max}(\mathbf{H})\left(\frac{|\mathcal{U}\cap\mathcal{B}|}{|\mathcal{U}|}\right)^2\frac{1}{|\mathcal{U}\cap\mathcal{B}|}\sum_{i\in(\mathcal{U}\cap\mathcal{B})}(\|\hat{\mathbf{p}}_i\|^2 + \|\mathbf{p}^*\|^2) \\
&\leq 4\sigma_{max}(\mathbf{H})\left(\frac{|\mathcal{U}\cap\mathcal{B}|}{|\mathcal{U}|}\right)^2\max_{i\in\mathcal{M}}(\|\hat{\mathbf{p}}_i\|^2 + \|\mathbf{p}^*\|^2) \\
&\leq 4\sigma_{max}(\mathbf{H})\left(\frac{|\mathcal{U}\cap\mathcal{B}|}{|\mathcal{U}|}\right)^2\max_{i\in\mathcal{M}}(\|\hat{\mathbf{p}}_i - \mathbf{p}^*\|^2 + 2\|\mathbf{p}^*\|^2) \\
&\leq 4\kappa\left(\frac{|\mathcal{U}\cap\mathcal{B}|}{|\mathcal{U}|}\right)^2\max_{i\in\mathcal{M}}(\|\mathbf{H}^{\frac{1}{2}}(\hat{\mathbf{p}}_i - \mathbf{p}^*)\|^2 + 2\|\mathbf{H}^{\frac{1}{2}}\mathbf{p}^*\|^2) \\
&\leq 4\kappa\left(\frac{|\mathcal{U}\cap\mathcal{B}|}{|\mathcal{U}|}\right)^2(\epsilon^2 + (2 + \zeta_1^2)\|\mathbf{H}^{\frac{1}{2}}\mathbf{p}^*\|^2) \\
&\leq 4\kappa\left(\frac{\alpha}{1-\beta}\right)^2(\epsilon^2 + (2 + \zeta_1^2)\|\mathbf{H}^{\frac{1}{2}}\mathbf{p}^*\|^2),
\end{aligned}$$

where $\zeta_1 = \nu\left(\eta + \frac{\eta^2}{1-\eta}\right) = \frac{\nu}{1-\eta}$ and $\kappa = \frac{\sigma_{max}(\mathbf{H})}{\sigma_{min}(\mathbf{H})}$.

Combining all the bounds on Term1 , Term2 and Term3 we have

$$\frac{1}{2} \|\mathbf{H}^{\frac{1}{2}}(\hat{\mathbf{p}} - \mathbf{p}^*)\|^2 \leq \epsilon_{byz}^2 + \zeta_{byz}^2 \|\mathbf{H}^{\frac{1}{2}} \mathbf{p}^*\|^2,$$

where

$$\begin{aligned} \epsilon_{byz}^2 &= \left(3 \left(\frac{1-\alpha}{1-\beta} \right)^2 + 4\kappa \left(\frac{\alpha}{1-\beta} \right)^2 \right) \epsilon^2, \\ \zeta_{byz}^2 &= 2 \left(\frac{1-\alpha}{1-\beta} \right)^2 \zeta_{\mathcal{M} \cap \mathcal{T}}^2 + \left(\frac{1-\alpha}{1-\beta} \right)^2 \zeta_{\mathcal{M}}^2 + 4\kappa \left(\frac{\alpha}{1-\beta} \right)^2 (2 + \zeta_1^2). \end{aligned}$$

Finally we have

$$\begin{aligned} \phi(\hat{\mathbf{p}}) - \phi(\mathbf{p}^*) &\leq \epsilon_{byz}^2 - \zeta_{byz}^2 \phi(\mathbf{p}^*) \\ \Rightarrow \phi(\hat{\mathbf{p}}) &\leq \epsilon_{byz}^2 + (1 - \zeta_{byz}^2) \phi(\mathbf{p}^*). \end{aligned}$$

□

Lemma 6.10. *Let $\zeta_{byz} \in (0, 1)$, ϵ_{byz} be any fixed parameter. And $\hat{\mathbf{p}}_t$ satisfies $\phi_t(\hat{\mathbf{p}}_t) \leq \epsilon_{byz}^2 + (1 - \zeta_{byz}^2) \min_{\mathbf{p}} \phi_t(\mathbf{p})$. Under the Assumption 6.1(Hessian L -Lipschitz) and $\Delta_t = \mathbf{w}_t - \mathbf{w}^*$ satisfies*

$$\Delta_{t+1}^T \mathbf{H}_t \Delta_{t+1} \leq L \|\Delta_{t+1}\| \|\Delta_t\|^2 + \frac{\zeta_{byz}^2}{1 - \zeta_{byz}^2} \Delta_t^T \mathbf{H}_t \Delta_t + 2\epsilon_{byz}^2.$$

Proof. We choose $\zeta = \zeta_{byz}$ and $\epsilon = \epsilon_{byz}$ from the Lemma 6.9 and follow the proof of Lemma 6.7 to obtain the desired bound. □

Proof of Theorem 6.8

Proof. We get the desired bound by developing from the result of the Lemma 6.10 and following the proof of Theorem 6.1 □

The remarks of Section 6.2 is also applicable here. On top of that, we have the following remarks:

Remark 6.5. *Compared to the convergence rate of Theorem 6.1, the rate here remains order-wise same even with Byzantine robustness. The coefficient of the quadratic term remains unchanged but the linear rate and the error floor suffers a little bit (by a small constant factor).*

Remark 6.6. *Note that for Theorem 6.8 to hold, we require $\alpha \sim 1/\sqrt{\kappa_t}$ for all t . In cases where κ_t is large, this can impose a stricter condition on α . However, we conjecture that this dependence can be improved via applying a more intricate (and perhaps computation heavy) Byzantine resilience algorithm. In this work, we kept the Byzantine resilience scheme simple at the expense of this condition on α .*

6.4 COMRADE Can Communicate Even Less and Resist Byzantine Workers

In Section 6.2 we analyze COMRADE with an additional feature. We let the worker machines further reduce the communication cost by applying a generic class of δ -approximate compressor [71] on the parameter update of Algorithm 4.

Worker machine i computes the product of local Hessian inverse and local gradient and then apply ρ -approximate compressor to obtain $\mathcal{Q}(\mathbf{H}_{i,t}^{-1}\mathbf{g}_{i,t})$; and finally sends this compressed vector to the center. The Byzantine resilience subroutine remains the same—except, instead of sorting with respect to $\|\mathbf{H}_{i,t}^{-1}\mathbf{g}_{i,t}\|$, the center machine now sorts according to $\|\mathcal{Q}(\mathbf{H}_{i,t}^{-1}\mathbf{g}_{i,t})\|$. The center machine aggregates the compressed updates by averaging $\mathcal{Q}(\hat{\mathbf{p}}) = \frac{1}{|\mathcal{U}_t|} \sum_{i \in \mathcal{U}_t} \mathcal{Q}(\hat{\mathbf{p}}_{i,t})$, and take the next step as $w_{t+1} = w_t - \gamma \mathcal{Q}(\hat{\mathbf{p}})$.

Recall the definition of ϵ from (6.7). We also use the following notation : $\zeta_{\mathcal{M}}^2 = \nu(\frac{\eta}{\sqrt{(1-\alpha)m}} + \frac{\eta^2}{1-\eta})$, $\zeta_1 = \frac{\nu}{1-\eta}$ and $\nu = \frac{\sigma_{\max}(\mathbf{A}^T \mathbf{A})}{\sigma_{\max}(\mathbf{A}^T \mathbf{A}) + n\lambda}$. Furthermore, we define the following:

$$\epsilon_{comp,byz}^2 = [3(\frac{1-\alpha}{1-\beta})^2 + 4\kappa_t(\frac{\alpha}{1-\beta})^2](1 + \kappa(1-\rho))\epsilon^2 \quad (6.16)$$

$$\begin{aligned} \zeta_{comp,byz}^2 &= 2(\frac{1-\alpha}{1-\beta})^2(\zeta_1^2 + \kappa_t(1-\rho)((1 + \zeta_1^2)) + (\frac{1-\alpha}{1-\beta})^2(\zeta_{\mathcal{M}}^2 + \kappa_t(1-\rho)((1 + \zeta_1^2))) \\ &\quad + 4\kappa_t(\frac{\alpha}{1-\beta})^2(2 + (\zeta_1^2 + \kappa_t(1-\rho)((1 + \zeta_1^2)))) \end{aligned} \quad (6.17)$$

Theorem 6.11. *Let $\mu \in [1, \frac{n}{d}]$ be the coherence of \mathbf{A}_t . Let $\gamma = 1$ and $s \geq \frac{3\mu d}{\eta^2} \log \frac{md}{\delta}$ for some $\eta, \delta \in (0, 1)$. For $0 \leq \alpha < \beta < 1/2$, under Assumption 6.1 and with \mathcal{Q} being the ρ -approximate compressor, with probability exceeding $1 - \delta$, we obtain*

$$\|\Delta_{t+1}\| \leq \max\left\{\sqrt{\kappa_t\left(\frac{\zeta_{comp,byz}^2}{1 - \zeta_{comp,byz}^2}\right)}\|\Delta_t\|, \frac{L}{\sigma_{\min}(\mathbf{H}_t)}\|\Delta_t\|^2\right\} + \frac{\epsilon_{comp,byz}}{\sqrt{\sigma_{\min}(\mathbf{H}_t)}}$$

where $\epsilon_{comp,byz}$ and $\zeta_{comp,byz}$ are given in equations (6.16) and (6.17) respectively.

Lemma 6.12. *Let $\{\mathbf{S}_i\}_{i=1}^m \in \mathbb{R}^{n \times s}$ be sketching matrices satisfies the Lemma 6.3. Let ϕ_t be defined in (6.5) and $\hat{\mathbf{p}}_t$ be defined in Algorithm 4. It holds that*

$$\min_{\mathbf{p}} \phi_t(\mathbf{p}) \leq \phi_t(\hat{\mathbf{p}}_t) \leq \epsilon_{byz}^2 + (1 - \alpha_{byz}^2)\phi(\mathbf{p}^*)$$

where

Proof. In the following analysis we omit the subscript 't'. From the definition of the quadratic function (6.5) we know that

$$\begin{aligned} \phi(\mathcal{Q}(\hat{\mathbf{p}})) - \phi(\mathbf{p}^*) &= \frac{1}{2}\|\mathbf{H}^{\frac{1}{2}}(\mathcal{Q}(\hat{\mathbf{p}}) - \mathbf{p}^*)\|^2 \\ &\leq \underbrace{\|\mathbf{H}^{\frac{1}{2}}(\mathcal{Q}(\hat{\mathbf{p}}) - \hat{\mathbf{p}})\|^2}_{Term1} + \underbrace{\|\mathbf{H}^{\frac{1}{2}}(\hat{\mathbf{p}} - \mathbf{p}^*)\|^2}_{Term2} \end{aligned}$$

First we bound the Term 1.

$$\begin{aligned}
\|\mathbf{H}^{\frac{1}{2}}(\mathcal{Q}(\hat{\mathbf{p}}) - \hat{\mathbf{p}})\|^2 &\leq \sigma_{\max}(\mathbf{H}_t)(1 - \rho)\|\hat{\mathbf{p}}\|^2 \\
&\leq \sigma_{\max}(\mathbf{H}_t)(1 - \rho)[\|\hat{\mathbf{p}} - \mathbf{p}^*\|^2 + \|\mathbf{p}^*\|^2] \\
&\leq \frac{\sigma_{\max}(\mathbf{H}_t)}{\sigma_{\min}(\mathbf{H}_t)}(1 - \rho)[\|\mathbf{H}^{\frac{1}{2}}(\hat{\mathbf{p}} - \mathbf{p}^*)\|^2 + \|\mathbf{H}^{\frac{1}{2}}\mathbf{p}^*\|^2] \\
&= \kappa(1 - \rho)[\|\mathbf{H}^{\frac{1}{2}}(\hat{\mathbf{p}} - \mathbf{p}^*)\|^2 + \|\mathbf{H}^{\frac{1}{2}}\mathbf{p}^*\|^2]
\end{aligned}$$

Now plugging back the bound of Term 1, we get

$$\begin{aligned}
\phi(\mathcal{Q}(\hat{\mathbf{p}})) - \phi(\mathbf{p}^*) &\leq \kappa(1 - \rho)[\|\mathbf{H}^{\frac{1}{2}}(\hat{\mathbf{p}} - \mathbf{p}^*)\|^2 + \|\mathbf{H}^{\frac{1}{2}}\mathbf{p}^*\|^2] + \|\mathbf{H}^{\frac{1}{2}}(\hat{\mathbf{p}} - \mathbf{p}^*)\|^2 \\
&= (1 + \kappa(1 - \rho))[\|\mathbf{H}^{\frac{1}{2}}(\hat{\mathbf{p}} - \mathbf{p}^*)\|^2] + \kappa(1 - \rho)\|\mathbf{H}^{\frac{1}{2}}\mathbf{p}^*\|^2
\end{aligned}$$

Now we use Lemma 6.6 to bound the term $\|\mathbf{H}^{\frac{1}{2}}(\hat{\mathbf{p}} - \mathbf{p}^*)\|^2$ and we get,

$$\begin{aligned}
\phi(\mathcal{Q}(\hat{\mathbf{p}})) - \phi(\mathbf{p}^*) &\leq (1 + \kappa(1 - \rho))(\epsilon^2 + \alpha^2\|\mathbf{H}^{\frac{1}{2}}\mathbf{p}^*\|^2) + \kappa(1 - \rho)\|\mathbf{H}^{\frac{1}{2}}\mathbf{p}^*\|^2 \\
&= (1 + \kappa(1 - \rho))\epsilon^2 + [(1 + \kappa(1 - \rho))\alpha^2 + \kappa(1 - \rho)]\|\mathbf{H}^{\frac{1}{2}}\mathbf{p}^*\|^2 \\
\Rightarrow \phi(\mathcal{Q}(\hat{\mathbf{p}})) &\leq (1 + \kappa(1 - \rho))\epsilon^2 + (1 - [(1 + \kappa(1 - \rho))\alpha^2 + \kappa(1 - \rho)])\phi(\mathbf{p}^*) \\
&= \epsilon_b^2 + (1 - \alpha_b^2)\phi(\mathbf{p}^*)
\end{aligned}$$

where

$$\begin{aligned}
\epsilon_b^2 &= (1 + \kappa(1 - \rho))\epsilon^2 \\
\alpha_b^2 &= (1 + \kappa(1 - \rho))\alpha^2 + \kappa(1 - \rho)
\end{aligned}$$

□

Lemma 6.13. $\Delta_t = \mathbf{w}_t - \mathbf{w}^*$ satisfies

$$\Delta_{t+1}^T \mathbf{H}_t \Delta_{t+1} \leq L \|\Delta_{t+1}\| \|\Delta_t\|^2 + \frac{\alpha_b^2}{1 - \alpha_b^2} \Delta_t^T \mathbf{H}_t \Delta_t + 2\epsilon_b^2$$

Theorem 6.14. Let $\mu_t \in [1, \frac{n}{d}]$ be the coherence of \mathbf{A}_t and m be the number of partitions. For some $\eta, \delta \in (0, 1)$, with probability $1 - \delta$

$$\|\Delta_{t+1}\| \leq \max\left\{\sqrt{\frac{\sigma_{\max}(\mathbf{H}_t)}{\sigma_{\min}(\mathbf{H}_t)}} \left(\frac{\alpha_b^2}{1 - \alpha_b^2} \|\Delta_t\|, \frac{L}{\sigma_{\min}(\mathbf{H}_t)} \|\Delta_t\|^2\right)\right\} + \frac{\epsilon_b}{\sqrt{\sigma_{\min}(\mathbf{H}_t)}}$$

Remark 6.7. With no compression ($\rho = 1$) we get back the convergence guarantee of Theorem 6.8.

Remark 6.8. Note that even with compression, we retain the linear-quadratic rate of convergence of COMRADE. The constants are affected by a ρ -dependent term.

6.5 Experimental Results

In this section we validate our algorithm, COMRADE in Byzantine and non-Byzantine setup on synthetically generated and benchmark LIBSVM [22] data-set. The experiments focus on the standard logistic regression problem. The logistic regression objective is defined as $\frac{1}{n} \sum_{i=1}^n \log(1 + \exp(-y_i \mathbf{x}_i^\top \mathbf{w})) + \frac{\lambda}{2n} \|\mathbf{w}\|^2$, where $\mathbf{w} \in \mathbb{R}^d$ is the parameter, $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^d$ are the feature data and $\{y_i\}_{i=1}^n \in \{0, 1\}$ are the corresponding labels. We use ‘mpi4py’ package in distributed computing framework (swarm2) at the University of Massachusetts Amherst [119]. We choose ‘a9a’ ($d = 123, n \approx 32K$), ‘w5a’ ($d = 300, n \approx 10k$), ‘Epsilon’ ($d = 2000, n = 0.4M$) and ‘covtype.binary’ ($d = 54, n \approx 0.5M$) classification datasets and partition the data in 20 different worker machines. In the experiments, we choose two types of Byzantine attacks : (1). ‘flipped label’-attack where (for binary classification) the Byzantine worker machines flip the labels of the data, thus making the model learn

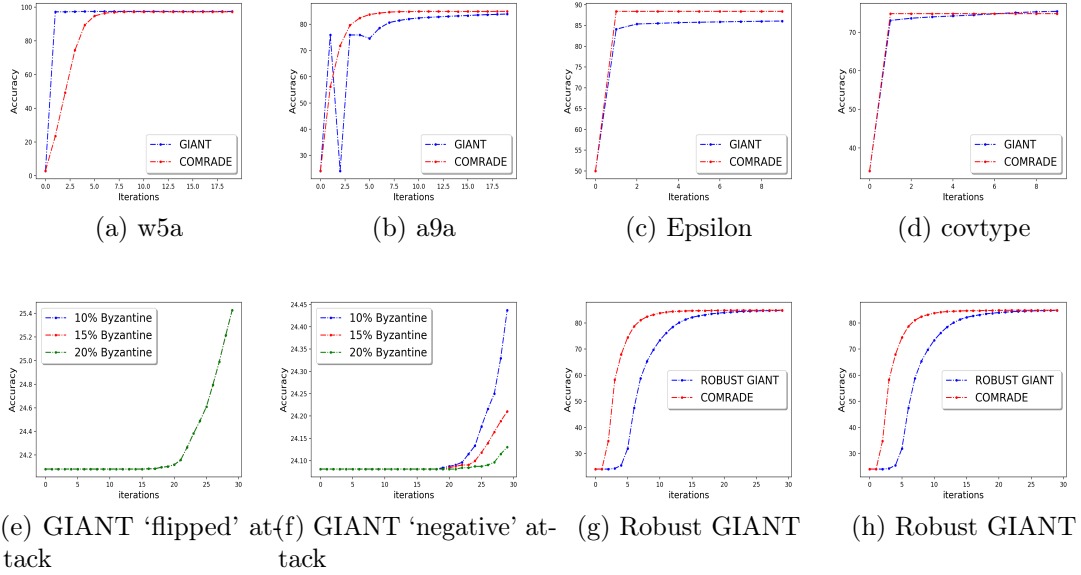


Figure 6.1: (First row) Comparison of training accuracy between COMRADE(Algorithm 4) and GIANT [127] with (a) w5a (b) a9a (c) Epsilon (d) Covtype dataset. (Second row) Training accuracy of (e) GIANT for ‘flipped label’ and (f) ‘negative update’ attack; and comparison of Robust GIANT and COMRADE with a9a dataset for (g) ‘flipped label’ and (h) ‘negative update’ attack.

with wrong labels, and (2). ‘negative update attack’ where the Byzantine worker machines compute the local update ($\hat{\mathbf{p}}_i$) and communicate $-c \times \hat{\mathbf{p}}_i$ with $c \in (0, 1)$ making the updates to be opposite of actual direction. We choose $\beta = \alpha + \frac{2}{m}$.

In Figure 6.1(first row) we compare COMRADE in non-Byzantine setup ($\alpha = \beta = 0$) with the state-of-the-art algorithm GIANT [127]. It is evident from the plot that despite the fact that COMRADE requires less communication, the algorithm is able to achieve similar accuracy. Also, we show the ineffectiveness of GIANT in the presence of Byzantine attacks. In Figure 6.2((e),(f)) we show the accuracy for flipped label and negative update attacks. These plots are an indicator of the requirement of robustness in the learning algorithm. So we devise ‘Robust GIANT’, which is GIANT algorithm with added ‘norm based thresholding’ for robustness. In particular, we trim the worker machines based on the local gradient norm in the first round of communication of GIANT. Subsequently, in the second round of communication, the non-trimmed

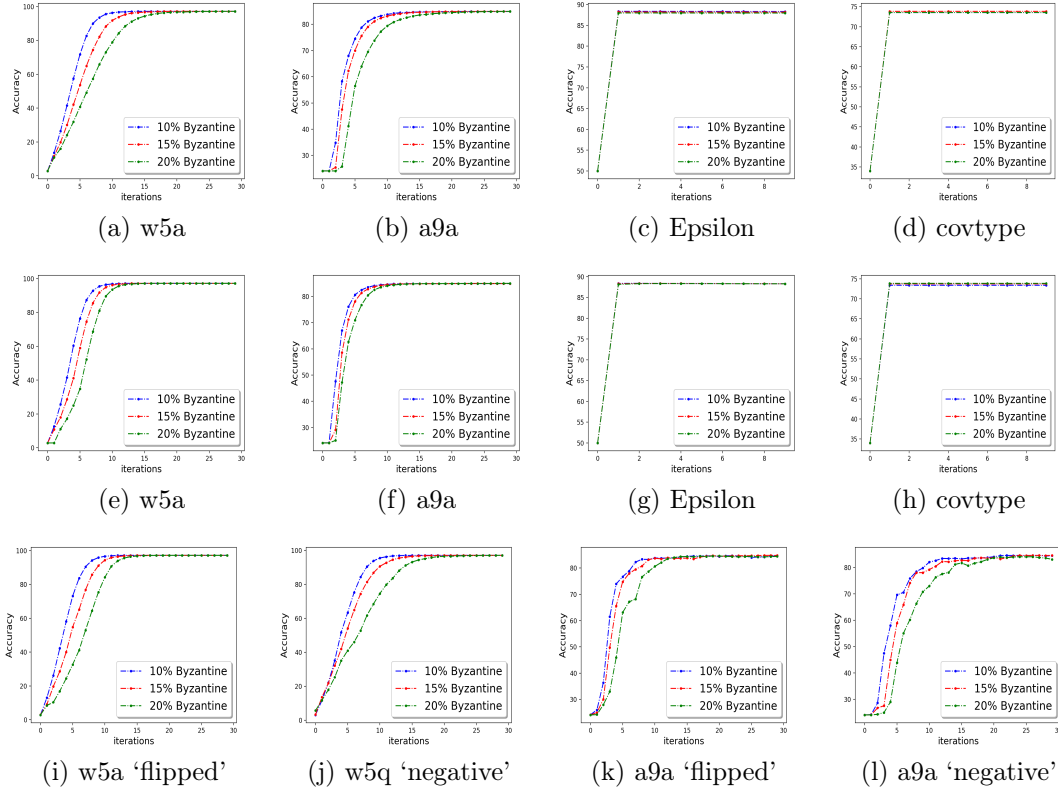


Figure 6.2: (First row) Accuracy of **COMRADE** with 10%, 15%, 20% Byzantine workers with ‘negative update’ attack for (a). w5a (b). a9a (c). covtype (d). Epsilon. (Second row) **COMRADE** accuracy with 10%, 15%, 20% Byzantine workers with ‘flipped label’ attack for (e) w5a (f) a9a (g) covtype (h) Epsilon. (Third row) Accuracy of **COMRADE** with ρ -approximate compressor (Section 6.4) with 10%, 15%, 20% Byzantine workers; (i) ‘flipped label’ attack for w5a (j) ‘negative update’ attack for w5a. (k) ‘flipped label’ attack for a9a . (l) ‘negative update’ attack for a9a dataset.

worker machines send the updates (product of local Hessian inverse and the local gradient) to the center machine. We compare **COMRADE** with ‘Robust GIANT’ in Figure 6.1((g),(h)) with 10% Byzantine worker machines for ‘a9a’ dataset. It is evident plot that **COMRADE** performs better than the ‘Robust GIANT’.

Next we show the accuracy of **COMRADE** with different numbers of Byzantine worker machines. Here we choose $c = 0.9$. We show the accuracy for ‘negative update’ attack in Figure 6.2(first row) and ‘flipped label’ attack in Figure 6.2 (second row). Furthermore, we show that **COMRADE** works even when ρ -approximate compressor is

applied to the updates. In Figure 6.2(Third row) we plot the training accuracies. For compression we apply the scheme known as QSGD [6].

6.6 Conclusion and Future Direction

In this chapter, we address the issue of the communication efficiency and Byzantine resilience for second order optimization. In the previous chapter (Chapter 5), we have already shown the use of compression for the second order method with *DINGO* as the underlying optimization algorithm. In contrast to that, we improve the algorithm *GIANT* in terms of communication cost with one round per iteration and eventually using compression on the update. We achieve the Byzantine resilience with norm based thresholding (also seen in Chapter 4).

As future work, the immediate interest would be to extend the ideas for general convex and non-convex loss function. The norm based thresholding is a very general purpose technique so the Byzantine resilience can be achieved. The challenge remain in the update and the deviation and the dissimilarity bound of the Hessian and the gradient. Due to the structure of the loss function in this chapter, the deviation bounds between the local and global Hessian and gradients are computed with assumption of incoherence in the data. For general loss function, dissimilarity assumption as seen in [69] is required. With these type of assumptions, the problem of general loss function and data heterogeneity of the data can be handled.

Even though the second order method has a faster convergence rate, it comes with an added cost of computation of the Hessian and inverse of the Hessian. It would be interesting to compute the computation cost in terms of number of gradients and Hessians that need to be computed in order to achieve certain level of convergence guarantee.

CHAPTER 7

ESCAPING SADDLE POINTS IN DISTRIBUTED NEWTON'S METHOD WITH BYZANTINE RESILIENCE

In this chapter, we continue with the m worker machines and center machine setup with α -fraction machines being Byzantine in nature. We consider the function $f(\cdot)$ to be non-convex. Our goal is to design algorithms which can avoid saddle points and find a local minima.

We consider a distributed variant of the cubic regularized Newton algorithm [94]. In this scheme, the center machine asks the workers to solve an auxiliary function and return the result. It is worth mentioning that in most distributed optimization paradigm, including Federated Learning, the workers possess sufficient compute power to handle this partial transfer of compute load, and in most cases, this is desirable [76]. The center machine aggregates the solution of the worker machines and takes a descent step. Note that, unlike gradient aggregation, the aggregation of the solutions of the local optimization problems is a highly non-linear operation. Hence, it is quite non-trivial to extend the centralized cubic regularized algorithm to a distributed one. The solution to the cubic regularization even lacks a closed form solution unlike the second order Hessian based update or the first order gradient based update. The analysis is carried out by leveraging the first order and second order stationary conditions of the auxiliary function solved in each worker machines.

A point \mathbf{x} is said to satisfy the ϵ -second order stationary condition of the loss function $f(\cdot)$ if,

$$\|\nabla f(\mathbf{x})\| \leq \epsilon \quad \lambda_{\min}(\nabla^2 f(\mathbf{x})) \geq -\sqrt{\epsilon}.$$

$\nabla f(\mathbf{x})$ denotes the gradient of the function and $\lambda_{\min}(\nabla^2 f(\mathbf{x}))$ denotes the minimum eigenvalue of the Hessian of the function. Hence, under the assumption (which is standard in the literature, see [64, 135]) that all saddle points are strict (i.e., $\lambda_{\min}(\nabla^2 f(\mathbf{x}_s)) < 0$ for any saddle point \mathbf{x}_s), all second order stationary points (with $\epsilon = 0$) are local minima, and hence converging to a stationary point is equivalent to converging to a local minima.

In addition to this, we use a simple norm-based thresholding approach to robustify the distributed cubic-regularized Newton method previously described in Chapter 4 and 6. However, since the local optimization problem lacks a closed form solution, using norm-based trimming is also technical challenging in this case. Handling the Byzantine worker machines becomes a bit more complicated as those stationary conditions of the good machines (non-Byzantine machine) do not hold for the Byzantine worker machines.

7.1 Problem Formulation

We minimize a loss function of the form

$$f(\mathbf{x}) = \frac{1}{m} \sum_{i=1}^m f_i(\mathbf{x}), \tag{7.1}$$

where the function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is twice differentiable and non-convex. In this work, we consider distributed optimization framework with m worker machines and one center machine where the worker machines communicate to the center machine. Each

worker machine is associated with a local loss function f_i . We assume that the data distribution is non-iid across workers, which is standard in frameworks like FL. In addition to this, we also consider the case where α fraction of the worker machines are Byzantine for some $\alpha < \frac{1}{2}$. The Byzantine machines can send arbitrary updates to the central machine which can disrupt the learning. Furthermore, the Byzantine machines can collude with each other, create *fake local minima* or attack maliciously by gaining information about the learning algorithm and other workers.

7.1.1 Our Contributions

We propose a novel distributed and robust cubic regularized Newton algorithm, that escapes saddle point efficiently. We prove that the algorithm convergence at a rate of $\frac{1}{T^{2/3}}$, which is faster than the first order methods (which converge at $1/\sqrt{T}$ rate, see [135]). Hence, the number of iterations (and hence the communication cost) required to achieve a target accuracy is much fewer than the first order methods. Furthermore, our algorithm plateaus at an error floor, which depends on the gradient and Hessian dissimilarity parameters (see Assumption 7.3 and 7.4). It is worth pointing out that in the presence of Byzantine workers, this error floor is unavoidable (see [135]). Moreover, several related literature [136, 56, 55] obtains similar error floor.

We address the saddle point avoidance problem in the presence of Byzantine workers. In particular, we use a simple norm-based thresholding scheme to resist Byzantine workers. In previous works, techniques like coordinate-wise median, coordinate-wise trimmed mean and spectral filtering are used to resist Byzantine workers.

In Section 7.5, we verify our theoretical findings via experiments. We use benchmark LIBSVM ([23]) datasets for logistic regression and non-convex robust regression and show convergence results for both non-Byzantine and several different Byzantine attacks. Specifically, we characterize the total iteration complexity (defined in Sec-

tion 7.5) of our algorithm, and compare it with the first order methods. We observe that the algorithm of [135] requires 25% more total iterations than ours.

7.2 Related Work

Saddle point avoidance algorithms In the recent years, there are handful first order algorithms [83, 82, 40] that focus on the escaping saddle points and convergence to local minima. The critical algorithmic aspect is running gradient based algorithm and adding perturbation to the iterates when the gradient is small. ByzantinePGD [135], PGD [64], Neon+GD[133], Neon2+GD [9] are examples of such algorithms. For faster convergence rate, second order Hessian based algorithms are developed for saddle point avoidance. The work of Nesterov and Polyak [94] first proposes the cubic regularized Newton method and provides analysis for the second order stationary condition. An algorithm called Adaptive Regularization with Cubics (ARC) was developed by [20, 21] where cubic regularized Newton method with access to inexact Hessian was studied. The inexactness of Hessians for the ARC algorithm is adaptive over iterations. Cubic regularization with both the gradient and Hessian being inexact was studied in [122]. In [73], a cubic regularized Newton with sub-sampled Hessian and gradient was proposed, but for analysis, the batch size of the sample changes in adaptive manner to provide guarantees for the inexactness of the Hessian and gradient. In this work, we also take a similar approach as [73], but we relax the adaptive nature of the sample size. Momentum based cubic regularized algorithm was studied in [129]. A variance reduced cubic regularized algorithm was proposed in [141, 128]. In terms of solving the cubic sub-problem, [19] proposes a gradient based algorithm and [4] provides a Hessian-vector product technique.

Byzantine resilience The effect of adversaries on convergence of non-convex optimization was studied in [33, 92]. In the distributed learning context, [?] proposes one shot median based robust learning. A median of mean based algorithm was proposed

in [27] where the worker machines are grouped in batches and the Byzantine resilience is achieved by computing the median of the grouped machines. Later [136] proposes co-ordinate wise median, trimmed mean and iterative filtering based approaches. Communication-efficient and Byzantine robust algorithms were developed in [13, 55]. A norm based thresholding approach for Byzantine resilience for distributed Newton algorithm was also developed [56]. All these works provide only first order convergence guarantee (small gradient). The work [135] is the only one that provides second order guarantee (Hessian positive semi-definite) under Byzantine attack.

Algorithm 5 Byzantine Robust Distributed Cubic Regularized Newton Algorithm

- 1: **Input:** Step size η_k , parameter $\beta \geq 0, \gamma > 0, M > 0$
 - 2: **Initialize:** Initial iterate $\mathbf{x}_0 \in \mathbb{R}^d$
 - 3: **for** $k = 0, 1, \dots, T - 1$ **do**
 - 4: Central machine: broadcasts \mathbf{x}_k
 for $i \in [m]$ **do in parallel**
 - 5: i -th worker machine:
 - Non-Byzantine: Computes local gradient $\mathbf{g}_{i,k}$ and local Hessian $\mathbf{H}_{i,k}$; locally solves the problem described in equation (7.2) and sends $\mathbf{s}_{i,k+1}$ to the central machine,
 - Byzantine: Generates \star (arbitrary), and sends it to the center machine
 - end for**
 - 6: Center Machine:
 - Sort the worker machines in a non decreasing order according to norm of updates $\{\mathbf{s}_{i,k+1}\}_{i=1}^m$ from the local machines
 - Return the indices of the first $1 - \beta$ fraction of machines as \mathcal{U}_t ,
 - Update parameter: $\mathbf{x}_{k+1} = \mathbf{x}_k + \eta_k \frac{1}{|\mathcal{U}_t|} \sum_{i \in \mathcal{U}_t} \mathbf{s}_{i,k+1}$
 - 7: **end for**
-

7.3 Distributed Cubic Regularized Newton

We first focus on the non-Byzantine setup ($\alpha = 0, \beta = 0$ in Algorithm 5) of distributed cubic regularized Newton algorithm. Byzantine resilience attribute of Algorithm 5 is deferred to Section 7.4. Let the local data at the i -th machine is

denoted by S_i . Starting with initialization \mathbf{x}_0 , the center machine broadcasts the parameter to the worker machines. At k -th iteration, the i -th worker machine solves a cubic-regularized auxiliary loss function based on its local data:

$$\mathbf{s}_{i,k+1} = \arg \min_{\mathbf{s}} \mathbf{g}_{i,k}^T \mathbf{s} + \frac{\gamma}{2} \mathbf{s}^T \mathbf{H}_{i,k} \mathbf{s} + \frac{M}{6} \gamma^2 \|\mathbf{s}\|^3, \quad (7.2)$$

where $M > 0, \gamma > 0$ are parameter and $\mathbf{g}_{i,t}, \mathbf{H}_{i,t}$ are the gradient and Hessian of the local loss function f_i computed on data (S_i) stored in the worker machine.

$$\mathbf{g}_{i,k} = \nabla f_i(\mathbf{x}_k) = \frac{1}{|S_i|} \sum_{\mathbf{z}_i \in S_i} \nabla f_i(\mathbf{x}_k, \mathbf{z}_i), \quad \mathbf{H}_{i,k} = \nabla^2 f_i(\mathbf{x}_k) = \frac{1}{|S_i|} \sum_{\mathbf{z}_i \in S_i} \nabla^2 f_i(\mathbf{x}_k, \mathbf{z}_i).$$

After receiving the update $\mathbf{s}_{i,k+1}$, the central machine updates the parameter in the following way

$$\mathbf{s}_{k+1} = \mathbf{s}_k + \frac{\eta_k}{m} \sum_{i=1}^m \mathbf{s}_{i,k+1}, \quad (7.3)$$

where η_k is the step-size.

Remark 7.1. *Note that, we introduce the parameter γ in the cubic regularized sub-problem. The parameter γ emphasizes the effect of the second and third order terms in the sub-problem. The choice of γ plays an important role in our analysis in handling the non-linear update from different worker machines. Such non-linearity vanishes if we choose $\gamma = 0$.*

7.3.1 Some Useful Facts

For the purpose of analysis we use the following sets of inequalities.

Fact 7.1. For a_1, \dots, a_n we have the following inequality

$$\left\| \left(\sum_{i=1}^n a_i \right) \right\|^3 \leq \left(\sum_{i=1}^n \|a_i\| \right)^3 \leq n^2 \sum_{i=1}^n \|a_i\|^3 \quad (7.4)$$

$$\left\| \left(\sum_{i=1}^n a_i \right) \right\|^2 \leq \left(\sum_{i=1}^n \|a_i\| \right)^2 \leq n \sum_{i=1}^n \|a_i\|^2 \quad (7.5)$$

Fact 7.2. For $a_1, \dots, a_n > 0$ and $r < s$

$$\left(\frac{1}{n} \sum_{i=1}^n a_i^r \right)^{1/r} \leq \left(\frac{1}{n} \sum_{i=1}^n a_i^s \right)^{1/s} \quad (7.6)$$

Lemma 7.3 ([94]). Under Assumption 7.2, i.e., the Hessian of the function is L_2 -Lipschitz continuous, for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, we have

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y}) - \nabla^2 f(\mathbf{x})(\mathbf{y} - \mathbf{x})\| \leq \frac{L_2}{2} \|\mathbf{y} - \mathbf{x}\|^2 \quad (7.7)$$

$$\left| f(\mathbf{y}) - f(\mathbf{x}) - \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) - \frac{1}{2}(\mathbf{y} - \mathbf{x})^T \nabla^2 f(\mathbf{x})(\mathbf{y} - \mathbf{x}) \right| \leq \frac{L_2}{6} \|\mathbf{y} - \mathbf{x}\|^2 \quad (7.8)$$

Next, we establish the following Lemma that provides some nice properties of the cubic sub-problem.

Lemma 7.4. Let $M > 0, \gamma > 0, \mathbf{g} \in \mathbb{R}^d, \mathbf{H} \in \mathbb{R}^{d \times d}$, and

$$\mathbf{s} = \arg \min_{\mathbf{x}} \mathbf{g}^T \mathbf{x} + \frac{\gamma}{2} \mathbf{x}^T \mathbf{H} \mathbf{x} + \frac{M\gamma^2}{6} \|\mathbf{x}\|^3. \quad (7.9)$$

The following holds

$$\mathbf{g} + \gamma \mathbf{H} \mathbf{s} + \frac{M\gamma^2}{2} \|\mathbf{s}\| \mathbf{s} = \mathbf{0}, \quad (7.10)$$

$$\mathbf{H} + \frac{M\gamma}{2} \|\mathbf{s}\| \mathbf{I} \succeq \mathbf{0}, \quad (7.11)$$

$$\mathbf{g}^T \mathbf{s} + \frac{\gamma}{2} \mathbf{s}^T \mathbf{H} \mathbf{s} \leq -\frac{M}{4} \gamma^2 \|\mathbf{s}\|^3. \quad (7.12)$$

Proof. The equations (7.10) and (7.11) are from the first and second order optimal condition. We proof (7.12), by using the conditions of (7.10) and (7.11).

$$\mathbf{g}^T \mathbf{s} + \frac{\gamma}{2} \gamma \mathbf{s}^T \mathbf{H} \mathbf{s} = - \left(\gamma \mathbf{H} \mathbf{s} + \frac{M}{2} \gamma^2 \|\mathbf{s}\| \mathbf{s} \right)^T \mathbf{s} + \frac{\gamma}{2} \gamma \mathbf{s}^T \mathbf{H} \mathbf{s} \quad (7.13)$$

$$\begin{aligned} &= -\gamma \mathbf{s}^T \mathbf{H} \mathbf{s} - \frac{M}{2} \gamma^2 \|\mathbf{s}\|^3 + \frac{\gamma}{2} \gamma \mathbf{s}^T \mathbf{H} \mathbf{s} \\ &\leq \frac{M}{4} \gamma^2 \|\mathbf{s}\|^3 - \frac{M}{2} \gamma^2 \|\mathbf{s}\|^3 \\ &= -\frac{M}{4} \gamma^2 \|\mathbf{s}\|^3. \end{aligned} \quad (7.14)$$

In (7.13), we substitute the expression \mathbf{g} from the equation (7.10). In (7.14), we use the fact that $\mathbf{s}^T \mathbf{H} \mathbf{s} + \frac{M\gamma}{2} \|\mathbf{s}\|^3 > 0$ from the equation (7.11). \square

7.3.2 Theoretical Guarantees

We have the following standard assumptions:

Assumption 7.1. *The non-convex loss function $f(\cdot)$ is twice continuously-differentiable and bounded below, i.e., $f^* = \inf_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x}) > -\infty$.*

Assumption 7.2. *The loss $f(\cdot)$ is L -Lipschitz continuous ($\forall \mathbf{x}, \mathbf{y}$, $|f(\mathbf{x}) - f(\mathbf{y})| \leq L \|\mathbf{x} - \mathbf{y}\|$), has L_1 -Lipschitz gradients ($\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L_1 \|\mathbf{x} - \mathbf{y}\|$) and L_2 -Lipschitz Hessian ($\|\nabla^2 f(\mathbf{x}) - \nabla^2 f(\mathbf{y})\| \leq L_2 \|\mathbf{x} - \mathbf{y}\|$).*

The above assumption states that the loss and the gradient and Hessian of the loss do not drastically change in the local neighborhood. These assumptions are standard in the analysis of the saddle point escape for cubic regularization (see [122, 73, 94, 19]).

In this work, we assume the data distribution across workers to be non-iid. However, we assume that the local gradient and Hessian computed at worker machines (using local data) satisfies the following gradient and Hessian dissimilarity conditions.

Assumption 7.3. *(Gradient dissimilarity) For a given $\epsilon_g > 0$ and for all k, i ,*

$$\|\nabla f(\mathbf{x}_k) - \mathbf{g}_{i,k}\| \leq \epsilon_g. \quad (7.15)$$

Assumption 7.4. (*Hessian dissimilarity*) For a given $\epsilon_H > 0$ and for all k, i ,

$$\|\nabla^2 f(\mathbf{x}_k) - \mathbf{H}_{i,k}\| \leq \epsilon_H. \quad (7.16)$$

Note that similar assumptions featured in previous literature. For example, in [68, 46], the authors use similar kind of dissimilarity assumptions that are prevalent in the *Federated learning* setup to highlight the *non-iid* or heterogeneity of the data among users.

Assumption 7.5. (*Bounded Space*) We assume that the domain of the solution of the sub-problem is bounded for all the workers and iterations.

$$\max_{i \in \mathcal{M}, k} \|\mathbf{s}_{i,k}\| \leq \Gamma$$

Remark 7.2. [*Values of ϵ_g and ϵ_H in special cases*] Compared to the Assumptions 7.3 and 7.4, the gradient and Hessian bound have been studied under more relaxed condition. In [73, 122, 129], the authors consider gradient and Hessian with sub-sampled data being drawn uniformly randomly from the data set. Due to the data being drawn in iid manner, both the bound (ϵ_g, ϵ_H) parameters value diminish at the rate $\propto 1/\sqrt{|S|}$ where $|S|$ is the size of the data sample in each worker machine. In [56], the authors analyze the deviation in case of data partitioning where each worker node sample data uniformly without replacement from a given data set.

Theorem 7.5. Under the Assumptions 7.1-7.5, and $\alpha = 0$, after T iterations, the sequence $\{\mathbf{x}_i\}_{i=1}^T$ generated by the Algorithm 4 with $\beta = 0$, contains a point $\tilde{\mathbf{x}}$ such that

$$\|\nabla f(\tilde{\mathbf{x}})\| \leq \frac{\Psi_1}{T^{\frac{2}{3}}} + \epsilon_g \quad \lambda_{\min}(\nabla^2 f(\tilde{\mathbf{x}})) \geq -\frac{\Psi_2}{T^{\frac{1}{3}}} - \epsilon_H, \quad (7.17)$$

where, $\lambda_{\min}(\cdot)$ denotes the minimum eigenvalue and

$$\begin{aligned}
\Psi_1 &= \left(\frac{L_2}{2} + \frac{M\gamma^2}{2\eta_k^2} \right) \Psi^2, \Psi_2 = \left(\frac{M\gamma}{2\eta_k} + L_2 \right) \Psi, \\
\lambda &= \left(\frac{M}{4m} - \frac{L_2}{6} \right) \\
\Psi &= \left[\frac{f(\mathbf{x}_0) - f^*}{\lambda} + \sum_{k=0}^{T-1} \frac{\lambda_\Gamma}{\lambda} \right]^{\frac{1}{3}}
\end{aligned}$$

Proof. Proof of Theorem 7.5

First we state the results of Lemma 7.4 for each worker node in iteration k ,

$$\mathbf{g}_{i,k} + \gamma \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} + \frac{M}{2} \gamma^2 \|\mathbf{s}_{i,k+1}\| \mathbf{s}_{i,k+1} = \mathbf{0} \quad (7.18)$$

$$\gamma \mathbf{H}_{i,k} + \frac{M}{2} \gamma^2 \|\mathbf{s}_{i,k+1}\| \mathbf{I} \succeq \mathbf{0} \quad (7.19)$$

$$\mathbf{g}_{i,k}^T \mathbf{s}_{i,k+1} + \frac{\gamma}{2} \mathbf{s}_{i,k+1}^T \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} \leq -\frac{M}{4} \gamma^2 \|\mathbf{s}_{i,k+1}\|^3 \quad (7.20)$$

At iteration k , we have,

$$\begin{aligned}
& f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) \\
& \leq \nabla f(\mathbf{x}_k)^T (\mathbf{x}_{k+1} - \mathbf{x}_k) + \frac{1}{2} (\mathbf{x}_{k+1} - \mathbf{x}_k)^T \nabla^2 f(\mathbf{x}_k) (\mathbf{x}_{k+1} - \mathbf{x}_k) + \frac{L_2}{6} \|\mathbf{x}_{k+1} - \mathbf{x}_k\|^3 \\
& \leq \eta_k \nabla f(\mathbf{x}_k)^T \mathbf{s}_{k+1} + \eta_k^2 \frac{1}{2} \mathbf{s}_{k+1}^T \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{k+1} + \frac{L_2}{6} \|\eta_k \mathbf{s}_{k+1}\|^3 \\
& = \frac{\eta_k}{m} \nabla f(\mathbf{x}_k)^T \left(\sum_i \mathbf{s}_{i,k+1} \right) + \frac{\eta_k^2}{2} \left(\frac{1}{m} \sum_i \mathbf{s}_{i,k+1} \right)^T \nabla^2 f(\mathbf{x}_k) \left(\frac{1}{m} \sum_i \mathbf{s}_{i,k+1} \right) + \frac{L_2 \eta_k^3}{6} \|\mathbf{s}_{k+1}\|^3 \\
& \leq \frac{\eta_k}{m} \left(\sum_i \nabla f(\mathbf{x}_k)^T \mathbf{s}_{i,k+1} \right) + \frac{\eta_k^2}{2m^2} \left(\sum_i \mathbf{s}_{i,k+1}^T \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{i,k+1} + \sum_{i \neq j} \mathbf{s}_{i,k+1}^T \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{j,k+1} \right) \\
& \quad + \frac{L_2 \eta_k^3}{6m} \sum_i \|\mathbf{s}_{i,k+1}\|^3 \\
& \leq \frac{\eta_k}{m} \sum_i \left(\mathbf{g}_{i,k}^T \mathbf{s}_{i,k+1} + \frac{\gamma}{2} \mathbf{s}_{i,k+1}^T \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} \right) + \frac{\eta_k}{m} \left(\sum_i (\nabla f(\mathbf{x}_k) - \mathbf{g}_{i,k+1})^T \mathbf{s}_{i,k+1} \right)
\end{aligned} \quad (7.21)$$

$$\begin{aligned}
& -\frac{\eta_k \gamma}{2m} \sum_i \mathbf{s}_{i,k+1}^T \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} \\
& + \frac{\eta_k^2}{2m^2} \left(\sum_i \mathbf{s}_{i,k+1}^T \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{i,k+1} + \sum_{i \neq j} \mathbf{s}_{i,k+1}^T \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{j,k+1} \right) + \frac{L_2 \eta_k^3}{6m} \sum_i \|\mathbf{s}_{i,k+1}\|^3 \\
& \leq \frac{\eta_k}{m} \sum_i -\frac{M}{4} \gamma^2 \|\mathbf{s}_{i,k+1}\|^3 + \frac{\eta_k}{m} \left(\sum_i \epsilon_g \|\mathbf{s}_{i,k+1}\| \right) - \left(\frac{\eta_k \gamma}{2m} - \frac{\eta_k^2}{2m^2} \right) \sum_i \mathbf{s}_{i,k+1}^T \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} \\
& + \frac{\eta_k^2}{2m^2} \left[\sum_i \mathbf{s}_{i,k+1}^T (\nabla^2 f(\mathbf{x}_k) - \mathbf{H}_{i,k}) \mathbf{s}_{i,k+1} + \sum_{i \neq j} \mathbf{s}_{i,k+1}^T \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{j,k+1} \right] + \frac{L_2 \eta_k^3}{6m} \sum_i \|\mathbf{s}_{i,k+1}\|^3
\end{aligned} \tag{7.22}$$

$$\begin{aligned}
& \leq \left(-\frac{M\gamma^2 \eta_k}{4m} + \frac{L_2 \eta_k^3}{6m} \right) \sum_i \|\mathbf{s}_{i,k+1}\|^3 + \frac{\eta_k \epsilon_g}{m} \left(\sum_i \|\mathbf{s}_{i,k+1}\| \right) + \left(\frac{\eta_k \gamma}{2m} - \frac{\eta_k^2}{2m^2} \right) \sum_i \frac{M}{2} \gamma \|\mathbf{s}_{i,k+1}\|^3 \\
& + \frac{\eta_k^2}{2m^2} \left[\epsilon_H \sum_i \|\mathbf{s}_{i,k+1}\|^2 + L_1 \sum_{i \neq j} \|\mathbf{s}_{i,k+1}\| \|\mathbf{s}_{j,k+1}\| \right]
\end{aligned} \tag{7.23}$$

$$\begin{aligned}
& = \left(-\frac{M\gamma^2 \eta_k}{4m^2} + \frac{L_2 \eta_k^3}{6m} \right) \sum_i \|\mathbf{s}_{i,k+1}\|^3 + \underbrace{\frac{\eta_k \epsilon_g}{m} \left(\sum_i \|\mathbf{s}_{i,k+1}\| \right)}_{\text{Term1}} \\
& + \underbrace{\frac{\eta_k^2}{2m^2} \left[\epsilon_H \sum_i \|\mathbf{s}_{i,k+1}\|^2 + L_1 \sum_{i \neq j} \|\mathbf{s}_{i,k+1}\| \|\mathbf{s}_{j,k+1}\| \right]}_{\text{Term2}}
\end{aligned} \tag{7.24}$$

In (7.21), we apply the inequality (7.14) on $\|\frac{1}{m} \sum_i \mathbf{s}_{i,k+1}\|^3$. In line (7.22), we use the gradient approximation from the Assumption 7.3. In line (7.22), we apply the fact that $\mathbf{s}_{i,k+1}^T \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} + \frac{M\gamma}{2} \|\mathbf{s}_{i,k+1}\|^3 > 0$ from the equation (7.19) and assume that $\gamma > \frac{\eta_k}{m}$. In line (7.23), we use the Assumption 7.4 and the fact that the Hessian of the objective function is bounded as the gradient is L_1 -Lipschitz continuous.

Now we bound the Term 1 and Term 2 in equation (7.24).

$$\text{Term 1} \leq \frac{\eta_k \epsilon_g}{m} \sum_i \|\mathbf{s}_{i,k+1}\| \leq \eta_k \epsilon_g \alpha \Gamma \tag{7.25}$$

Now we focus on Term 2,

$$\begin{aligned}
\text{Term 2} &\leq \frac{\eta_k^2}{2m^2} \left[\epsilon_H \sum_i \|\mathbf{s}_{i,k+1}\|^2 + L_1 \left(\left\| \sum_i \mathbf{s}_{i,k+1} \right\|^2 - \sum_i \|\mathbf{s}_{i,k+1}\|^2 \right) \right] \\
&\leq \frac{\eta_k^2}{2m^2} ((m-1)L_1 + \epsilon_H) \sum_i \|\mathbf{s}_{i,k+1}\|^2
\end{aligned} \tag{7.26}$$

$$\leq \frac{\eta_k^2}{2m} ((m-1)L_1 + \epsilon_H) \alpha \Gamma^2 \tag{7.27}$$

Combining the result of (7.25) and (7.27) in equation (7.24), we have

$$\begin{aligned}
f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) &\leq \left[-\frac{M\gamma^2\eta_k}{4m^2} + \frac{L_2\eta_k^3}{6m} \right] \sum_i \|\mathbf{s}_{i,k+1}\|^3 + \frac{\eta_k^2}{2m} ((m-1)L_1 + \epsilon_H) \alpha \Gamma^2 + \eta_k \epsilon_g \alpha \Gamma \\
&\leq -\lambda \frac{1}{m} \sum_i \|\mathbf{s}_{i,k+1}\|^3 + \lambda_\Gamma
\end{aligned}$$

Now we consider that $\lambda = \left(\frac{M\gamma}{4\eta_k m} - \frac{L_2}{6} \right)$. We can assure $\lambda > 0$ by choosing $M \geq \frac{4\eta_k m}{\gamma} \frac{L_2}{6}$. And we have $\lambda_\Gamma = \frac{\eta_k^2}{2m} ((m-1)L_1 + \epsilon_H) \alpha \Gamma^2 + \eta_k \epsilon_g \alpha \Gamma$.

Now we have

$$\frac{1}{m} \sum_i \|\eta_k \mathbf{s}_{i,k+1}\|^3 \leq \frac{1}{\lambda} [f(\mathbf{x}_k) - f(\mathbf{x}_{k+1}) + \lambda_\Gamma]$$

Now we consider the step k_0 , where $k_0 = \arg \min_{0 \leq k \leq T-1} \|\mathbf{x}_{k+1} - \mathbf{x}_k\| = \arg \min_{0 \leq k \leq T-1} \|\eta_k \mathbf{s}_{k+1}\|$.

$$\begin{aligned}
\min_{0 \leq k \leq T} \|\mathbf{x}_{k+1} - \mathbf{x}_k\|^3 &= \min_{0 \leq k \leq T} \|\eta_k \mathbf{s}_{k+1}\|^3 \\
&\leq \frac{1}{m} \sum_{i=1}^m \|\eta_{k_0} \mathbf{s}_{i,k_0+1}\|^3 \\
&\leq \frac{1}{T} \sum_{k=0}^{T-1} \frac{1}{m} \sum_{i=1}^m \|\eta_k \mathbf{s}_{i,k+1}\|^3 \\
&\leq \frac{1}{T} \sum_{k=0}^{T-1} \frac{1}{\lambda} [f(\mathbf{x}_k) - f(\mathbf{x}_{k+1}) + \lambda_\Gamma] \\
&= \frac{1}{T} \left[\frac{f(\mathbf{x}_0) - f(\mathbf{x}_T)}{\lambda} + \lambda_\Gamma \right] \\
&\leq \frac{1}{T} \left[\frac{f(\mathbf{x}_0) - f^*}{\lambda} + \sum_{k=0}^{T-1} \frac{\lambda_\Gamma}{\lambda} \right]
\end{aligned}$$

With the choice of $\eta_k = \frac{c}{T}$ and $\gamma = \frac{c_1}{T}$ we have $\lambda_\Gamma = O(1/T)$ and $\lambda = O(1)$. So the sum is actually constant. Based on the calculation above, we have

$$\begin{aligned}\|\mathbf{x}_{k_0+1} - \mathbf{x}_{k_0}\| &\leq \left(\frac{1}{m} \sum_{i=1}^m \|\eta_{k_0} \mathbf{s}_{i,k_0+1}\|^3 \right)^{\frac{1}{3}} \\ &\leq \frac{1}{T^{\frac{1}{3}}} \left[\frac{f(\mathbf{x}_0) - f^*}{\lambda} + \sum_{k=0}^{T-1} \frac{\lambda_\Gamma}{\lambda} \right]^{\frac{1}{3}}\end{aligned}$$

We choose $\Psi = \left[\frac{f(\mathbf{x}_0) - f^*}{\lambda} + \sum_{k=0}^{T-1} \frac{\lambda_\Gamma}{\lambda} \right]^{\frac{1}{3}}$ and have

$$\|\mathbf{x}_{k_0+1} - \mathbf{x}_{k_0}\| \leq \left(\frac{1}{m} \sum_{i=1}^m \|\eta_{k_0} \mathbf{s}_{i,k_0+1}\|^3 \right)^{\frac{1}{3}} \leq \frac{\Psi}{T^{\frac{1}{3}}} \quad (7.28)$$

Now the gradient condition

$$\begin{aligned}&\|\nabla f(\mathbf{x}_{k+1})\| \\ &= \left\| \nabla f(\mathbf{x}_{k+1}) - \frac{1}{m} \sum_{i=1}^m \mathbf{g}_{i,k} - \frac{1}{m} \sum_{i=1}^m \left(\gamma \mathbf{H}_{i,k+1} \mathbf{s}_{i,k+1} - \frac{M\gamma^2}{2} \|\mathbf{s}_{i,k+1}\| \mathbf{s}_{i,k+1} \right) \right\| \quad (7.29) \\ &\leq \left\| \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k) - \nabla^2 f(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) \right\| + \left\| \frac{1}{m} \sum_{i=1}^m (\mathbf{g}_{i,k} - \nabla f(\mathbf{x}_k)) \right\| \\ &\quad + \left\| \nabla^2 f(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) - \gamma \frac{1}{m} \sum_{i=1}^m \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} \right\| + \left\| \frac{1}{m} \sum_{i=1}^m \frac{M\gamma^2}{2} \|\mathbf{s}_{i,k+1}\| \mathbf{s}_{i,k+1} \right\| \\ &\leq \frac{L_2}{2} \|\eta_k \mathbf{s}_{k+1}\|^2 + \left\| \frac{\eta_k}{m} \sum_{i=1}^m \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{i,k+1} - \frac{\gamma}{m} \sum_{i=1}^m \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} \right\| + \frac{M\gamma^2}{2m} \sum_i \|\mathbf{s}_{i,k+1}\|^2 + \epsilon_g \quad (7.30)\end{aligned}$$

$$\begin{aligned}&\leq \left(\frac{L_2 \eta_k^2}{2m} + \frac{M\gamma^2}{2m} \right) \sum_i \|\mathbf{s}_{i,k+1}\|^2 + \left\| \frac{\eta_k}{m} \sum_{i=1}^m \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{i,k+1} - \frac{\gamma}{m} \sum_{i=1}^m \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{i,k+1} \right\| \\ &\quad + \left\| \frac{\gamma}{m} \sum_{i=1}^m \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{i,k+1} - \frac{\gamma}{m} \sum_{i=1}^m \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} \right\| + \epsilon_g \\ &\leq \left(\frac{L_2 \eta_k^2}{2m} + \frac{M\gamma^2}{2m} \right) \sum_i \|\mathbf{s}_{i,k+1}\|^2 + \frac{(\eta_k - \gamma)L_1}{m} \sum_i \|\mathbf{s}_{i,k+1}\| + \frac{\gamma\epsilon_H}{m} \sum_i \|\mathbf{s}_{i,k+1}\| + \epsilon_g \quad (7.31)\end{aligned}$$

$$\leq \left(\frac{L_2 \eta_k^2}{2} + \frac{M\gamma^2}{2} \right) \frac{1}{m} \sum_i \|\mathbf{s}_{i,k+1}\|^2 + (\eta_k L_1 - \gamma(L_1 - \epsilon_H)) \frac{1}{m} \sum_i \|\mathbf{s}_{i,k+1}\| + \epsilon_g \quad (7.32)$$

We choose $\gamma > \eta_k \frac{L_1}{L_1 - \epsilon_H}$. So we have

$$\|\nabla f(\mathbf{x}_{k+1})\| \leq \left(\frac{L_2}{2} + \frac{M\gamma^2}{2\eta_k^2} \right) \frac{1}{m} \sum_i \|\eta_k \mathbf{s}_{i,k+1}\|^2 + \epsilon_g$$

At step k_0 , by choosing $\eta_k = \gamma$, we have

$$\|\nabla f(\mathbf{x}_{k_0+1})\| \leq \left(\frac{L_2}{2} + \frac{M\gamma^2}{2\eta_k^2} \right) \frac{\Psi^2}{T^{\frac{2}{3}}} + \epsilon_g \quad (7.33)$$

where, $\Psi_1 = \left(\frac{L_2}{2} + \frac{M\gamma^2}{2\eta_k^2} \right) \Psi^2$. The Hessian bound

$$\begin{aligned} & \lambda_{\min}(\nabla^2 f(\mathbf{x}_{k+1})) \\ &= \frac{1}{m} \sum_{i=1}^m \lambda_{\min} [\nabla^2 f(\mathbf{x}_{k+1})] \\ &= \frac{1}{m} \sum_{i=1}^m \lambda_{\min} [\mathbf{H}_{i,k} - (\mathbf{H}_{i,k} - \nabla^2 f(\mathbf{x}_{k+1}))] \\ &\geq \frac{1}{m} \sum_{i=1}^m [\lambda_{\min}(\mathbf{H}_{i,k}) - \|\mathbf{H}_{i,k} - \nabla^2 f(\mathbf{x}_{k+1})\|] \\ &\geq \frac{1}{m} \sum_{i=1}^m \lambda_{\min}(\mathbf{H}_{i,k}) - \frac{1}{m} \sum_{i=1}^m \|\mathbf{H}_{i,k} - \nabla^2 f(\mathbf{x}_{k+1})\| \end{aligned} \quad (7.34)$$

$$\begin{aligned} &\geq \frac{1}{m} \sum_{i=1}^m -\frac{M\gamma}{2} \|\mathbf{s}_{i,k+1}\| - \frac{1}{m} \sum_{i=1}^m \|\mathbf{H}_{i,k} - \nabla^2 f(\mathbf{x}_k)\| - \frac{1}{m} \sum_{i=1}^m \|\nabla^2 f(\mathbf{x}_k) - \nabla^2 f(\mathbf{x}_{k+1})\| \\ &\geq \frac{1}{m} \sum_{i=1}^m -\frac{M\gamma}{2} \|\mathbf{s}_{i,k+1}\| - \epsilon_H - \frac{1}{m} \sum_{i=1}^m \|\nabla^2 f(\mathbf{x}_k) - \nabla^2 f(\mathbf{x}_{k+1})\| \\ &\geq \frac{1}{m} \sum_{i=1}^m -\frac{M\gamma}{2} \|\mathbf{s}_{i,k+1}\| - \epsilon_H - \frac{1}{m} \sum_{i=1}^m L_2 \|\mathbf{x}_k - \mathbf{x}_{k+1}\| \\ &\geq \left(-\frac{M\gamma}{2\eta_k} - L_2 \right) \frac{1}{m} \sum_{i=1}^m \|\eta_k \mathbf{s}_{i,k+1}\| - \epsilon_H \end{aligned} \quad (7.35)$$

$$\geq -\left(\frac{M\gamma}{2\eta_k} + L_2\right) \left(\frac{1}{m} \sum_{i=1}^m \|\eta_k \mathbf{s}_{i,k+1}\|^3\right)^{1/3} - \epsilon_H \quad (7.36)$$

Equation (7.34) follows from Weyl's inequality. We apply the Hessian approximation from the Assumption 7.4 in equation (7.35). In equation (7.36), we apply the result described in (7.6).

At step k_0 , by choosing $\eta_k = \gamma$, we have

$$\begin{aligned} \lambda_{\min}(\nabla^2 f(\mathbf{x}_{k_0+1})) &\leq -\left(\frac{M}{2} + L_2\right) \frac{\Psi_{\frac{1}{3}}}{T^{\frac{1}{3}}} - \epsilon_H \\ &= -\frac{\Psi_2}{T^{\frac{1}{3}}} - \epsilon_H \end{aligned} \quad (7.37)$$

where, $\Psi_2 = \left(\frac{M}{2} + L_2\right) \Psi^{1/3}$.

□

Remark 7.3. We choose the step sizes $\{\eta_k\}_{k=0}^{T-1}$ such way that $\sum_{i=0}^T \eta_k$ and $\sum_{i=0}^T \eta_k^2$ is bounded. For the ease of choice, we can choose $\eta_k = \frac{c}{T}$, for some constant $c > 0$.

Remark 7.4. Both the gradient and the minimum eigenvalue of the Hessian in the Theorem 7.5 have two parts. The first part decreases with the number iterations T . The gradient and the minimum eigenvalue of the Hessian have the rate of $O(1/T^{\frac{2}{3}})$ and $O(1/T^{\frac{1}{3}})$, respectively. Both of these rates match the rates of the centralized version of the cubic regularized Newton. In the second parts of the gradient bound and the minimum eigenvalue of the Hessian have the error floor of ϵ_g and ϵ_H , respectively. As mentioned above (see Remark 7.2), in the special cases, both the terms ϵ_g and ϵ_H decrease at the rate of $1/\sqrt{|S|}$, where $|S|$ is the number of data in each of the worker machines.

Remark 7.5 (Two rounds of communication $\epsilon_g = 0$). We can improve the bound in the Theorem 7.5, with the calculation of the actual gradient which requires one more round of communication in each iteration. In the first iteration, all the worker

machines compute the gradient based on the stored data and send it to the center machine. The center machine averages them and then broadcast the global gradient $\nabla f(\mathbf{x}_k) = \frac{1}{m} \sum_{i=1}^m \mathbf{g}_{i,k}$ at iteration k . In this manner, the worker machines solve the sub-problem (7.2) with the actual gradient. The analysis follows same as that of the Theorem 7.5 with $\epsilon_g = 0$. This improves the gradient bound while the communication remains $O(d)$ in each iteration.

7.3.3 Solution of the Cubic Sub-Problem

The cubic regularized sub-problem (7.2) needs to be solved to update the parameter. As this particular problem does not have a closed form solution, a solver is usually employed which yields a satisfactory solution. In previous works, different types of solvers have been used. [20, 21] solve the sub-problem using Lanczos based method in Krylov subspace. In [4], the authors propose a solver based on Hessian-vector product and binary search. Gradient descent based solver is proposed in [19, 122].

In the previous works, for example [129, 141, 128], the authors consider the exact solution of the cubic sub-problem for theoretical analysis. Recently, inexact solutions to the sub-problem is also proposed in the centralized (non-distributed) framework. For instance, [73] analyzes the cubic model with sub-sampled Hessian with approximate model minimization technique developed in [20]. Moreover, [122] shows improved analysis with gradient based minimization which is a variant studied in [19]. Both exact and inexact solutions to the sub-problem yields similar theoretical guarantees.

In our framework, each worker machine is tasked with solving the sub-problem. For the purpose of theoretical convergence analysis, we consider that worker machines obtain the exact solution in each round. However, in experiments (Section 7.5), we apply the gradient based solver of [122] to solve the sub-problem. Here, we let each worker machines run the gradient based solver for 10 iterations and send the update to the center machine in each iteration. Furthermore, , we provide an analysis where the

sub-problem is solved *approximately*, and characterize how the approximation error affects the convergence rate of Algorithm 4.

We consider that the solution to the sub-problem described in equation (7.2) to be inexact. The solver returns $\mathbf{s}_{i,k+1} + \Delta_{i,k+1}$ instead of the global minimizer $\mathbf{s}_{i,k+1}$ for i th worker in iteration k . In the following we show the consequence of approximate solution. The update at the center machine in iteration k is

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \eta_k \frac{1}{m} \sum_{i=1}^m \tilde{\mathbf{s}}_{i,k+1} = \mathbf{x}_k + \eta_k (\mathbf{s}_{k+1} + \Delta_{k+1})$$

Here $\Delta_{k+1} = \frac{1}{m} \sum_{i=1}^m \Delta_{i,k+1}$. This is the inexact part in the update in the iteration $k+1$. We define an uniform bound on Δ_{k+1} for all iterations as $\|\Delta_{k+1}\| \leq \|\Delta\|$.

$$\begin{aligned} & f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) \\ & \leq \nabla f(\mathbf{x}_k)^T (\mathbf{x}_{k+1} - \mathbf{x}_k) + \frac{1}{2} (\mathbf{x}_{k+1} - \mathbf{x}_k)^T \nabla^2 f(\mathbf{x}_k) (\mathbf{x}_{k+1} - \mathbf{x}_k) + \frac{L_2}{6} \|\mathbf{x}_{k+1} - \mathbf{x}_k\|^3 \\ & \leq \eta_k \nabla f(\mathbf{x}_k)^T (\mathbf{s}_{k+1} + \Delta_{k+1}) + \eta_k^2 \frac{1}{2} (\mathbf{s}_{k+1} + \Delta_{k+1})^T \nabla^2 f(\mathbf{x}_k) (\mathbf{s}_{k+1} + \Delta_{k+1}) + \frac{L_2}{6} \|\eta_k (\mathbf{s}_{k+1} + \Delta_{k+1})\|^3 \\ & \leq \underbrace{\eta_k \nabla f(\mathbf{x}_k)^T \mathbf{s}_{k+1} + \eta_k^2 \frac{1}{2} (\mathbf{s}_{k+1}^T \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{k+1}) + \frac{c_1 L_2}{6} \|\eta_k \mathbf{s}_{k+1}\|^3}_{Term1} \\ & \quad + \underbrace{\eta_k \nabla f(\mathbf{x}_k)^T \Delta_{k+1} + \frac{\eta_k^2}{2} (2 \mathbf{s}_{k+1}^T \nabla^2 f(\mathbf{x}_k) \Delta_{k+1} + \Delta_{k+1}^T \nabla^2 f(\mathbf{x}_k) \Delta_{k+1}) + \frac{c_1 L_2}{6} \|\eta_k \Delta_{k+1}\|^3}_{Term2} \end{aligned}$$

In the above c_1 is a constant. Now we bound Term 2

$$Term\ 2 \leq \eta_k \left[\epsilon_g + \frac{\eta_k}{2} (2 \|\mathbf{s}_{k+1}\| \epsilon_H \|\Delta_{k+1}\|) + \frac{c_1 L_2 \eta_k^2}{6} \|\Delta\|^2 \right] \|\Delta_{k+1}\| \leq \eta_k C_2 \|\Delta\|$$

Term 1 can be bounded in the line of the proof of Theorem 7.5 and we can achieve,

$$f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) \leq -\lambda \frac{1}{m} \sum_i \|\mathbf{s}_{i,k+1}\|^3 + \lambda_\Gamma + \eta_k C_2 \|\Delta\|$$

And for step k_0 , we get the following

$$\left(\frac{1}{m} \sum_{i=1}^m \|\eta_{k_0} \mathbf{s}_{i,k_0+1}\|^3 \right)^{\frac{1}{3}} \leq \frac{\Psi}{T^{\frac{1}{3}}}$$

where $\Psi = \left[\frac{f(\mathbf{x}_0) - f^*}{\lambda} + \sum_{k=0}^{T-1} \frac{\lambda_\Gamma}{\lambda} \right]^{\frac{1}{3}}$

Now the gradient condition

$$\begin{aligned} & \|\nabla f(\mathbf{x}_{k+1})\| \\ &= \left\| \nabla f(\mathbf{x}_{k+1}) - \frac{1}{m} \sum_{i=1}^m \mathbf{g}_{i,k} - \frac{1}{m} \sum_{i=1}^m \left(\gamma \mathbf{H}_{i,k+1} \mathbf{s}_{i,k+1} - \frac{M\gamma^2}{2} \|\mathbf{s}_{i,k+1}\| \mathbf{s}_{i,k+1} \right) \right\| \\ &\leq \left\| \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k) - \nabla^2 f(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) \right\| + \left\| \frac{1}{m} \sum_{i=1}^m (\mathbf{g}_{i,k} - \nabla f(\mathbf{x}_k)) \right\| \\ &\quad + \left\| \nabla^2 f(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) - \gamma \frac{1}{m} \sum_{i=1}^m \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} \right\| + \left\| \frac{1}{m} \sum_{i=1}^m \frac{M\gamma^2}{2} \|\mathbf{s}_{i,k+1}\| \mathbf{s}_{i,k+1} \right\| \\ &\leq L_2 \|\eta_k \mathbf{s}_{k+1}\|^2 + \left\| \frac{\eta_k}{m} \sum_{i=1}^m \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{i,k+1} - \frac{\gamma}{m} \sum_{i=1}^m \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} \right\| + \frac{M\gamma^2}{2m} \sum_i \|\mathbf{s}_{i,k+1}\|^2 + \epsilon_g \\ &\quad L_2 \eta_k^2 \|\Delta\|^2 + \|\nabla^2 f(\mathbf{x}_k)\| \|\Delta\| \end{aligned}$$

Following the gradient bound of the proof of the Theorem 7.5, we have

$$\|\nabla f(\mathbf{x}_{k+1})\| \leq \|\nabla f(\mathbf{x}_{k_0+1})\| \leq \left(\frac{L_2}{2} + \frac{M\gamma^2}{2\eta_k^2} \right) \frac{\Psi^2}{T^{\frac{2}{3}}} + \epsilon_g + (L_2 \eta_k \|\Delta\| + \epsilon_H) \|\Delta\|$$

For the Hessian bound, we use the (7.35),

$$\begin{aligned} \lambda_{\min}(\nabla^2 f(\mathbf{x}_{k+1})) &\geq \frac{1}{m} \sum_{i=1}^m -\frac{M\gamma}{2} \|\mathbf{s}_{i,k+1}\| - \epsilon_H - \frac{1}{m} \sum_{i=1}^m \|\nabla^2 f(\mathbf{x}_k) - \nabla^2 f(\mathbf{x}_{k+1})\| \\ &\geq \frac{1}{m} \sum_{i=1}^m -\frac{M\gamma}{2} \|\mathbf{s}_{i,k+1}\| - \epsilon_H - \frac{1}{m} \sum_{i=1}^m L_2 \|\mathbf{x}_k - \mathbf{x}_{k+1}\| \\ &\geq \left(-\frac{M\gamma}{2\eta_k} - L_2 \right) \frac{1}{m} \sum_{i=1}^m \|\eta_k \mathbf{s}_{i,k+1}\| - \epsilon_H - L_2 \|\Delta\| \end{aligned}$$

$$\geq -\left(\frac{M\gamma}{2\eta_k} + L_2\right) \left(\frac{1}{m} \sum_{i=1}^m \|\eta_k \mathbf{s}_{i,k+1}\|^3\right)^{1/3} - \epsilon_H - L_2 \|\Delta\|$$

Now comparing with the results of the Theorem 7.5, the gradient and Hessian bounds suffer only by additional term $(L_2\eta_k\|\Delta\| + \epsilon_H)\|\Delta\|$ and $L_2\|\Delta\|$ respectively. This additional terms are of the order $O(\|\Delta\|)$. If the worker machines solve the sub-problem exactly ($\|\Delta\| = 0$), we get back the guarantees described in Theorem 7.5.

7.4 Byzantine Resilience

In this section, we analyze our algorithm's resilience against Byzantine workers. We consider that $\alpha(< \frac{1}{2})$ fraction of the worker machines are Byzantine in nature. We denote the set of Byzantine worker machines by \mathcal{B} and the set of the rest of the good machines as \mathcal{M} . In each iteration, the good machines send the solution of the sub-cubic problem described in equation (7.2) and the Byzantine machines can send any arbitrary values or intentionally disrupt the learning algorithm with malicious updates. Moreover, in the non-convex optimization problems, one of the more complicated and important issue is to avoid saddle points which can yield highly sub-optimal results. In the presence of Byzantine worker machines, they can be in cohort to create a *fake local minima* and drive the algorithm into sub-optimal region. Lack of any robust measure towards these type of intentional and unintentional attacks can be catastrophic to the learning procedure as the learning algorithm can get stuck in such sub-optimal point. To tackle such Byzantine worker machines, we employ a simple process called *norm based thresholding*.

After receiving all the updates from the worker machines, the central machine outputs a set \mathcal{U} which consists of the indexes of the worker machines with smallest norm. We choose the size of the set \mathcal{U} to be $(1 - \beta)m$. Hence, we 'trim' β fraction of the worker machine so that we can control the iterated update by not letting the

worker machines with large norm participate and diverge the learning process. We denote the set of trimmed machine as \mathcal{T} . We choose $\beta > \alpha$ so that at least one of the good machines gets trimmed. In this way, the norm of the all the updates in the set \mathcal{U} is bounded by at least the largest norm of the good machines.

Theorem 7.6. *Suppose for non-Byzantine machines, Assumptions 7.1-7.4 hold and $0 \leq \alpha \leq \beta \leq \frac{1}{2}$. Running Algorithm 5 for T iterations, the sequence $\{\mathbf{x}_i\}_{i=1}^T$ generated contains a point $\tilde{\mathbf{x}}$ such that*

$$\|\nabla f(\tilde{\mathbf{x}})\| \leq \frac{\Psi_{1,byz}}{T^{\frac{2}{3}}} + \epsilon_g + \text{Minor Terms } O\left(\frac{1}{T}\right); \lambda_{\min}(\nabla^2 f(\tilde{\mathbf{x}})) \geq -\frac{\Psi_{2,byz}}{T^{\frac{1}{3}}} - \epsilon_H. \quad (7.38)$$

Proof. Proof of Theorem 7.6 We consider the following

$$\begin{aligned} & f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) \\ & \leq \nabla f(\mathbf{x}_k)^T (\mathbf{x}_{k+1} - \mathbf{x}_k) + \frac{1}{2} (\mathbf{x}_{k+1} - \mathbf{x}_k)^T \nabla^2 f(\mathbf{x}_k) (\mathbf{x}_{k+1} - \mathbf{x}_k) + \frac{L_2}{6} \|\mathbf{x}_{k+1} - \mathbf{x}_k\|^3 \\ & = \underbrace{\frac{\eta_k}{|\mathcal{U}|} \nabla f(\mathbf{x}_k)^T \sum_{i \in \mathcal{U}} \mathbf{s}_{i,k+1}}_{Term1} + \underbrace{\frac{\eta_k^2}{2|\mathcal{U}|^2} \left(\sum_{i \in \mathcal{U}} \mathbf{s}_{i,k+1} \right)^T \nabla^2 f(\mathbf{x}_k) \left(\sum_{i \in \mathcal{U}} \mathbf{s}_{i,k+1} \right)}_{Term2} + \underbrace{\frac{L_2}{6} \left\| \frac{\eta_k}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} \mathbf{s}_{i,k+1} \right\|^3}_{Term3} \end{aligned} \quad (7.39)$$

In line (7.39), we expand the update $\mathbf{x}_{k+1} - \mathbf{x}_k = \frac{\eta_k}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} \mathbf{s}_{i,k+1}$. Also we use the following fact.

$$|\mathcal{U}| = |\mathcal{U} \cap \mathcal{M}| + |\mathcal{U} \cap \mathcal{B}| \quad (7.40)$$

$$|\mathcal{M}| = |\mathcal{U} \cap \mathcal{M}| + |\mathcal{T} \cap \mathcal{M}| \quad (7.41)$$

Combining both the equations (7.40) and (7.41), we have

$$|\mathcal{U}| = |\mathcal{M}| - |\mathcal{T} \cap \mathcal{M}| + |\mathcal{U} \cap \mathcal{B}| \quad (7.42)$$

We use the fact of (7.42) to bound each term in equation (7.39). First, consider the Term 1,

$$\begin{aligned}
& \frac{\eta_k}{|\mathcal{U}|} \nabla f(\mathbf{x}_k)^T \sum_{i \in \mathcal{U}} \mathbf{s}_{i,k+1} \\
&= \frac{\eta_k}{(1-\beta)m} \nabla f(\mathbf{x}_k)^T \left[\sum_{i \in \mathcal{M}} \mathbf{s}_{i,k+1} - \sum_{i \in \mathcal{M} \cap \mathcal{T}} \mathbf{s}_{i,k+1} + \sum_{i \in \mathcal{U} \cap \mathcal{B}} \mathbf{s}_{i,k+1} \right] \\
&= \frac{\eta_k}{(1-\beta)m} \left[\sum_{i \in \mathcal{M}} \mathbf{g}_{i,k}^T \mathbf{s}_{i,k+1} - \sum_{i \in \mathcal{M} \cap \mathcal{T}} \nabla f(\mathbf{x}_k)^T \mathbf{s}_{i,k+1} + \sum_{i \in \mathcal{U} \cap \mathcal{B}} \nabla f(\mathbf{x}_k)^T \mathbf{s}_{i,k+1} + \frac{\gamma}{2} \sum_{i \in \mathcal{M}} \mathbf{s}_{i,k+1}^T \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} \right] \\
&\quad - \frac{\eta_k \gamma}{2(1-\beta)m} \sum_{i \in \mathcal{M}} \mathbf{s}_{i,k+1}^T \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} + \frac{\eta_k}{(1-\beta)m} \sum_{i \in \mathcal{M}} (\nabla f(\mathbf{x}_k) - \mathbf{g}_{i,k})^T \mathbf{s}_{i,k+1}
\end{aligned} \tag{7.43}$$

$$\begin{aligned}
&\leq -\frac{M\gamma^2\eta_k}{4(1-\beta)m} \left[\sum_{i \in \mathcal{M}} \|\mathbf{s}_{i,k+1}\|^3 \right] + \frac{\eta_k \epsilon_g}{(1-\beta)m} \sum_{i \in \mathcal{M}} \|\mathbf{s}_{i,k+1}\| - \frac{\eta_k \gamma}{2(1-\beta)m} \sum_{i \in \mathcal{M}} \mathbf{s}_{i,k+1}^T \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} \\
&\quad + \frac{\eta_k L}{(1-\beta)m} \left(\sum_{i \in \mathcal{M} \cap \mathcal{T}} \|\mathbf{s}_{i,k+1}\| + \sum_{i \in \mathcal{U} \cap \mathcal{B}} \|\mathbf{s}_{i,k+1}\| \right) \\
&\leq -\frac{M\gamma^2\eta_k}{4(1-\beta)m} \left[\sum_{i \in \mathcal{M}} \|\mathbf{s}_{i,k+1}\|^3 \right] - \frac{\eta_k \gamma}{2(1-\beta)m} \sum_{i \in \mathcal{M}} \mathbf{s}_{i,k+1}^T \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} + \frac{\eta_k}{(1-\beta)} ((1-\alpha)\epsilon_g + L)\Gamma
\end{aligned} \tag{7.44}$$

We use the following facts in (7.44).

- $\mathbf{g}_{i,k}^T \mathbf{s}_{i,k+1} + \frac{\gamma}{2} \mathbf{s}_{i,k+1}^T \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} \leq -\frac{M}{4} \gamma^2 \|\mathbf{s}_{i,k+1}\|^3$ and sum over the set \mathcal{M} .
- The gradient approximation described in Assumption 7.3.
- As the function f is L -Lipschitz, the gradient is bounded.

Now we bound Term 3 as follows,

$$\begin{aligned}
\frac{L_2}{6} \left\| \frac{\eta_k}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} \mathbf{s}_{i,k+1} \right\|^3 &\leq \frac{L_2 \eta_k^3}{6(1-\beta)m} \sum_{i \in \mathcal{U}} \|\mathbf{s}_{i,k+1}\|^3 \\
&\leq \frac{L_2 \eta_k^3}{6(1-\beta)m} \left[\sum_{i \in \mathcal{M}} \|\mathbf{s}_{i,k+1}\|^3 - \sum_{i \in \mathcal{M} \cap \mathcal{T}} \|\mathbf{s}_{i,k+1}\|^3 + \sum_{i \in \mathcal{U} \cap \mathcal{B}} \|\mathbf{s}_{i,k+1}\|^3 \right]
\end{aligned} \tag{7.45}$$

$$\leq \frac{L_2 \eta_k^3}{6(1-\beta)m} \left[\sum_{i \in \mathcal{M}} \|\mathbf{s}_{i,k+1}\|^3 + \alpha m \Gamma^3 \right] \quad (7.46)$$

In line (7.45), we use the inequality described in (7.4) and in line (7.46), we split the sum using (7.42).

Finally, we bound Term 2

$$\begin{aligned} & \frac{\eta_k^2}{2|\mathcal{U}|^2} \left(\sum_{i \in \mathcal{U}} \mathbf{s}_{i,k+1} \right)^T \nabla^2 f(\mathbf{x}_k) \left(\sum_{i \in \mathcal{U}} \mathbf{s}_{i,k+1} \right) \\ &= \frac{\eta_k^2}{2(1-\beta)^2 m^2} \left(\sum_{i \in \mathcal{U}} \mathbf{s}_{i,k+1}^T \nabla f(\mathbf{x}_k) \mathbf{s}_{i,k+1} + \sum_{i \neq j \in \mathcal{U}} \mathbf{s}_{i,k+1}^T \nabla f(\mathbf{x}_k) \mathbf{s}_{j,k+1} \right) \\ &= \frac{\eta_k^2}{2(1-\beta)^2 m^2} \\ & \quad \times \left(\sum_{i \in \mathcal{M}} \mathbf{s}_{i,k+1}^T (\nabla^2 f(\mathbf{x}_k) - \mathbf{H}_{i,k}) \mathbf{s}_{i,k+1} - \sum_{i \in \mathcal{M} \cap \mathcal{T}} \mathbf{s}_{i,k+1}^T \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{i,k+1} + \sum_{i \in \mathcal{U} \cap \mathcal{B}} \mathbf{s}_{i,k+1}^T \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{i,k+1} \right) \\ & \quad + \frac{\eta_k^2}{2(1-\beta)^2 m^2} \sum_{i \neq j \in \mathcal{U}} \mathbf{s}_{i,k+1}^T \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{j,k+1} + \frac{\eta_k^2}{2(1-\beta)^2 m^2} \sum_{i \in \mathcal{M}} \mathbf{s}_{i,k+1}^T \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} \end{aligned} \quad (7.47)$$

$$\begin{aligned} & \leq \frac{\eta_k^2}{2(1-\beta)^2 m^2} \left[\epsilon_H \sum_{i \in \mathcal{M}} \|\mathbf{s}_{i,k+1}\|^2 + L_1 \sum_{i \in \mathcal{M} \cap \mathcal{T}} \|\mathbf{s}_{i,k+1}\|^2 + L_1 \sum_{i \in \mathcal{U} \cap \mathcal{B}} \|\mathbf{s}_{i,k+1}\|^2 \right] \\ & \quad + \frac{\eta_k^2}{2(1-\beta)^2 m^2} \left[L_1 \left\| \sum_{i \in \mathcal{U}} \mathbf{s}_{i,k+1} \right\|^2 - L_1 \sum_{i \in \mathcal{U}} \|\mathbf{s}_{i,k+1}\|^2 \right] + \frac{\eta_k^2}{2(1-\beta)^2 m^2} \sum_{i \in \mathcal{M}} \mathbf{s}_{i,k+1}^T \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} \end{aligned} \quad (7.48)$$

$$\begin{aligned} & \leq \frac{\eta_k^2}{2(1-\beta)^2 m} [\epsilon_H(1-\alpha) + L_1] \Gamma^2 + \frac{\eta_k^2}{2(1-\beta)^2 m^2} \sum_{i \in \mathcal{M}} \mathbf{s}_{i,k+1}^T \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} \\ & \quad + \frac{\eta_k^2 L_1}{2(1-\beta)^2 m} ((1-\beta)m - 1) \Gamma^2 \end{aligned} \quad (7.49)$$

Collecting all the results we have

$$\begin{aligned} & f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) \\ & \leq -\frac{M\gamma^2\eta_k}{4(1-\beta)m} \left[\sum_{i \in \mathcal{M}} \|\mathbf{s}_{i,k+1}\|^3 \right] - \frac{\eta_k\gamma}{2(1-\beta)m} \left(\gamma - \frac{\eta_k}{(1-\beta)m} \right) \sum_{i \in \mathcal{M}} \mathbf{s}_{i,k+1}^T \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} \end{aligned}$$

$$\begin{aligned}
& + \frac{\eta_k}{(1-\beta)}((1-\alpha)\epsilon_g + L)\Gamma + \frac{L_2\eta_k^3}{6(1-\beta)m} \left[\sum_{i \in \mathcal{M}} \|\mathbf{s}_{i,k+1}\|^3 + \alpha m \Gamma^3 \right] \\
& + \frac{\eta_k^2}{2(1-\beta)^2 m} [\epsilon_H(1-\alpha) + (1-\beta)mL_1] \Gamma^2 \\
\leq & - \left[\frac{M\gamma\eta_k^2}{4(1-\beta)m^2} - \frac{L_2\eta_k^3}{6(1-\beta)m} \right] \sum_{i \in \mathcal{M}} \|\mathbf{s}_{i,k+1}\|^3 + \frac{L_2\eta_k^3}{6(1-\beta)} \alpha \Gamma^3 \\
& + \frac{\eta_k^2}{2(1-\beta)^2 m} [\epsilon_H(1-\alpha) + (1-\beta)mL_1] \Gamma^2 + \frac{\eta_k}{(1-\beta)}((1-\alpha)\epsilon_g + L)\Gamma \\
= & - \lambda_{byz} \frac{1}{(1-\alpha)m} \sum_{i \in \mathcal{M}} \|\eta_k \mathbf{s}_{i,k+1}\|^3 + \lambda_{floor}
\end{aligned}$$

where

$$\lambda_{byz} = \left[\frac{M\gamma}{4(1-\beta)m} - \frac{L_2}{6(1-\beta)} \right] (1-\alpha)$$

Now we can have the following results from the proof of Theorem 7.5 for step

$$k_0 = \arg \min_{0 \leq k \leq T-1} \|\mathbf{x}_{k+1} - \mathbf{x}_k\|$$

$$\begin{aligned}
\|\mathbf{x}_{k_0+1} - \mathbf{x}_{k_0}\|^3 & \leq \left[\frac{1}{(1-\beta)m} \sum_{i \in \mathcal{U}} \|\eta_{k_0} \mathbf{s}_{i,k_0+1}\|^3 \right] \\
& \leq \frac{(1-\alpha)}{(1-\beta)} \left[\frac{1}{(1-\alpha)m} \sum_{i \in \mathcal{M}} \|\eta_{k_0} \mathbf{s}_{i,k_0+1}\|^3 + \frac{\alpha}{1-\alpha} \eta_{k_0}^3 \Gamma^3 \right] \\
& \leq \frac{(1-\alpha)}{(1-\beta)} \frac{1}{T} \left[\frac{f(\mathbf{x}_0) - f^*}{\lambda_{byz}} + \frac{\sum_{k=0}^{T-1} \lambda_{floor}}{\lambda_{byz}} + \frac{\alpha}{1-\alpha} \eta_{k_0}^3 \Gamma^3 \right] \quad (7.50)
\end{aligned}$$

$$= \frac{\Psi_\Gamma}{T} \quad (7.51)$$

where, $\Psi_{byz} = \left[\frac{f(\mathbf{x}_0) - f^*}{\lambda_{byz}} + \frac{\sum_{k=0}^{T-1} \lambda_{floor}}{\lambda_{byz}} \right]^{\frac{1}{3}}$ and $\psi_\Gamma = \frac{(1-\alpha)}{(1-\beta)} \left[\frac{f(\mathbf{x}_0) - f^*}{\lambda_{byz}} + \frac{\sum_{k=0}^{T-1} \lambda_{floor}}{\lambda_{byz}} + \frac{\alpha}{1-\alpha} \eta_{k_0}^3 \Gamma^3 \right]$

The gradient condition

$$\begin{aligned}
& \|\nabla f(\mathbf{x}_{k+1})\| \\
= & \left\| \nabla f(\mathbf{x}_{k+1}) - \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathbf{g}_{i,k} - \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \gamma \mathbf{H}_{i,k+1} \mathbf{s}_{i,k+1} - \frac{M\gamma^2}{2} \|\mathbf{s}_{i,k+1}\| \mathbf{s}_{i,k+1} \right\| \quad (7.52)
\end{aligned}$$

$$\begin{aligned}
&\leq \left\| \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k) - \nabla^2 f(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) \right\| + \left\| \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} (\mathbf{g}_{i,k} - \nabla f(\mathbf{x}_k)) \right\| \\
&\quad + \left\| \nabla^2 f(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) - \gamma \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} \right\| + \left\| \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \frac{M\gamma^2}{2} \|\mathbf{s}_{i,k+1}\| \mathbf{s}_{i,k+1} \right\| \\
&\leq \frac{L_2 \eta_k^2}{2} \left\| \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} \mathbf{s}_{i,k+1} \right\|^2 + \epsilon_g + \frac{M\gamma^2}{2} \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \|\mathbf{s}_{i,k+1}\|^2 \\
&\quad + \left\| \frac{\eta_k}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{i,k+1} - \frac{\gamma}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{i,k+1} \right\| \\
&\quad + \left\| \frac{\gamma}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{i,k+1} - \frac{\gamma}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} \right\| \tag{7.53}
\end{aligned}$$

In line (7.52), we use the fact the first order optimal condition (7.18) holds for the good machines in the set \mathcal{M} . And in (7.53), we use the in exact gradient condition from Assumption 7.3 and the condition (7.20). Consider the term

$$\begin{aligned}
&\left\| \frac{\eta_k}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{i,k+1} - \frac{\gamma}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{i,k+1} \right\| \\
&\quad + \left\| \frac{\gamma}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{i,k+1} - \frac{\gamma}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathbf{H}_{i,k} \mathbf{s}_{i,k+1} \right\| \\
&= \left\| \frac{\eta_k}{(1-\beta)m} \left[\sum_{i \in \mathcal{M}} \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{i,k+1} - \frac{\gamma}{(1-\alpha)m} \sum_{i \in \mathcal{M}} \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{i,k+1} \right] \right\| \\
&\quad + \frac{\eta_k}{(1-\beta)m} \left(\left\| \sum_{i \in \mathcal{M} \cap \mathcal{T}} \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{i,k+1} \right\| + \left\| \sum_{i \in \mathcal{B} \cap \mathcal{U}} \nabla^2 f(\mathbf{x}_k) \mathbf{s}_{i,k+1} \right\| \right) + \frac{\gamma}{(1-\alpha)m} \epsilon_H \|\mathbf{s}_{i,k+1}\| \\
&\leq \left(\frac{\eta_k}{(1-\beta)m} - \frac{\gamma}{(1-\alpha)m} \right) L_1 \sum_{i \in \mathcal{M}} \|\mathbf{s}_{i,k+1}\| + \frac{\eta_k L_1}{(1-\beta)} \|\mathbf{s}_{i,k+1}\| + \frac{\gamma}{(1-\alpha)m} \epsilon_H \sum_{i \in \mathcal{M}} \|\mathbf{s}_{i,k+1}\| \\
&= \left(\frac{\eta_k L_1}{(1-\beta)m} - \frac{\gamma(L_1 - \epsilon_H)}{(1-\alpha)m} \right) \sum_{i \in \mathcal{M}} \|\mathbf{s}_{i,k+1}\| + \frac{\eta_k L_1}{(1-\beta)} \|\mathbf{s}_{i,k+1}\| \\
&\leq \frac{\eta_k L_1}{(1-\beta)} \Gamma \tag{7.54}
\end{aligned}$$

We choose $\gamma > \frac{\eta_k(1-\alpha)}{(1-\beta)} \frac{L_1}{L_1 - \epsilon_H}$ in equation (7.54). Now we have,

$$\frac{L_2 \eta_k^2}{2} \left\| \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} \mathbf{s}_{i,k+1} \right\|^2 \leq \frac{L_2 \eta_k^2}{2(1-\beta)m} \sum_{i \in \mathcal{U}} \|\mathbf{s}_{i,k+1}\|^2$$

$$\begin{aligned}
&\leq \frac{L_2 \eta_k^2}{2(1-\beta)m} \left[\sum_{i \in \mathcal{M}} \|\mathbf{s}_{i,k+1}\|^2 + \sum_{i \in \mathcal{B} \cap \mathcal{T}} \|\mathbf{s}_{i,k+1}\|^2 \right] \\
&\leq \frac{L_2}{2(1-\beta)m} \sum_{i \in \mathcal{M}} \|\eta_k \mathbf{s}_{i,k+1}\|^2 + \frac{L_2 \eta_k^2}{2(1-\beta)} \alpha \Gamma^2 \quad (7.55)
\end{aligned}$$

Putting the calculation of (7.54) and (7.55), in (7.53), we have,

$$\begin{aligned}
&\|\nabla f(\mathbf{x}_{k+1})\| \\
&\leq \left(\frac{L_2 \eta_k^2}{2(1-\beta)m} + \frac{M\gamma^2}{2(1-\alpha)m} \right) \sum_{i \in \mathcal{M}} \|\mathbf{s}_{i,k+1}\|^2 + \epsilon_g + \frac{\gamma \epsilon_H}{(1-\alpha)} \Gamma + \frac{L_2 \eta_k^2}{2(1-\beta)} \alpha \Gamma^2 \\
&\leq \left(\frac{L_2(1-\alpha)}{2(1-\beta)} + \frac{M\gamma^2}{2\eta_k^2} \right) \frac{1}{(1-\alpha)m} \sum_{i \in \mathcal{M}} \|\eta_k \mathbf{s}_{i,k+1}\|^2 + \epsilon_g + \frac{\gamma \epsilon_H}{(1-\alpha)} \Gamma + \frac{L_2 \eta_k^2}{2(1-\beta)} \alpha \Gamma^2 \\
&\leq \left(\frac{L_2(1-\alpha)}{2(1-\beta)} + \frac{M\gamma^2}{2\eta_k^2} \right) \left[\frac{1}{(1-\alpha)m} \sum_{i \in \mathcal{M}} \|\eta_k \mathbf{s}_{i,k+1}\|^3 \right]^{2/3} + \epsilon_g + \frac{\gamma \epsilon_H}{(1-\alpha)} \Gamma + \frac{L_2 \eta_k^2}{2(1-\beta)} \alpha \Gamma^2 \\
&\leq \left(\frac{L_2(1-\alpha)}{2(1-\beta)} + \frac{M\gamma^2}{2\eta_k^2} \right) \left(\frac{\psi_{byz}}{T} \right)^{2/3} + \epsilon_g + \frac{\gamma \epsilon_H}{(1-\alpha)} \Gamma + \frac{L_2 \eta_k^2}{2(1-\beta)} \alpha \Gamma^2 \quad (7.56)
\end{aligned}$$

We use the power mean inequality described in (7.6) in line (7.56). Then at step k_0 , we have,

$$\|\nabla f(\mathbf{x}_{k_0+1})\| \leq \frac{\Psi_{1,byz}}{T^{2/3}} + \epsilon_g + \text{Minor term } O\left(\frac{1}{T}\right), \quad (7.57)$$

where

$$\Psi_{1,byz} = \left(\frac{L_2(1-\alpha)}{2(1-\beta)} + \frac{M\gamma^2}{2\eta_k^2} \right) (\psi_{byz})^{2/3}$$

The Hessian bound is

$$\begin{aligned}
&\lambda_{\min}(\nabla^2 f(\mathbf{x}_{k+1})) \\
&= \frac{1}{(1-\alpha)m} \sum_{i \in \mathcal{M}} \lambda_{\min} [\nabla^2 f(\mathbf{x}_{k+1})]
\end{aligned}$$

$$\begin{aligned}
&= \frac{1}{(1-\alpha)m} \sum_{i \in \mathcal{M}} \lambda_{\min} [\mathbf{H}_{i,k} - (\mathbf{H}_{i,k} - \nabla^2 f(\mathbf{x}_{k+1}))] \\
&\geq \frac{1}{(1-\alpha)m} \sum_{i \in \mathcal{M}} [\lambda_{\min}(\mathbf{H}_{i,k}) - \|\mathbf{H}_{i,k} - \nabla^2 f(\mathbf{x}_{k+1})\|] \tag{7.58}
\end{aligned}$$

$$\begin{aligned}
&\geq \frac{1}{(1-\alpha)m} \sum_{i \in \mathcal{M}} \lambda_{\min}(\mathbf{H}_{i,k}) - \frac{1}{(1-\alpha)m} \sum_{i \in \mathcal{M}} \|\mathbf{H}_{i,k} - \nabla^2 f(\mathbf{x}_{k+1})\| \\
&\geq \frac{1}{(1-\alpha)m} \sum_{i \in \mathcal{M}} -\frac{M\gamma}{2} \|\mathbf{s}_{i,k+1}\| - \frac{1}{(1-\alpha)m} \sum_{i \in \mathcal{M}} \|\mathbf{H}_{i,k} - \nabla^2 f(\mathbf{x}_k)\| \\
&\quad - \frac{1}{(1-\alpha)m} \sum_{i \in \mathcal{M}} \|\nabla^2 f(\mathbf{x}_k) - \nabla^2 f(\mathbf{x}_{k+1})\| \tag{7.59} \\
&\geq \frac{1}{(1-\alpha)m} \sum_{i \in \mathcal{M}} -\frac{M\gamma}{2} \|\mathbf{s}_{i,k+1}\| - \epsilon_H - \frac{1}{(1-\alpha)m} \sum_{i \in \mathcal{M}} L_2 \|\mathbf{x}_k - \mathbf{x}_{k+1}\| \\
&= -\frac{M\gamma}{2\eta_k} \frac{1}{(1-\alpha)m} \sum_{i \in \mathcal{M}} \|\eta_k \mathbf{s}_{i,k+1}\| - L_2 \|\mathbf{x}_k - \mathbf{x}_{k+1}\| - \epsilon_H \\
&\geq -\frac{M\gamma}{2\eta_k} \left[\frac{1}{(1-\alpha)m} \sum_{i \in \mathcal{M}} \|\eta_k \mathbf{s}_{i,k+1}\|^3 \right]^{1/3} - L_2 \|\mathbf{x}_k - \mathbf{x}_{k+1}\| - \epsilon_H \tag{7.60}
\end{aligned}$$

In (7.58), we use the Weyl's inequality. In (7.60), we use the power mean inequality described in (7.6). At step k_0 , we have

$$\begin{aligned}
\lambda_{\min}(\nabla^2 f(\mathbf{x}_{k_0+1})) &\geq -\frac{M\gamma}{2\eta_k} \left(\frac{\psi_{byz}}{T}\right)^{1/3} - L_2 \left(\frac{\psi_\Gamma}{T}\right)^{1/3} - \epsilon_H \\
&= -\left(\frac{M\gamma}{2\eta_k} \psi_{byz}^{1/3} + L_2 \psi_\Gamma^{1/3}\right) \frac{1}{T^{1/3}} - \epsilon_H \\
&= -\frac{\psi_{2,byz}}{T^{1/3}} - \epsilon_H \tag{7.61}
\end{aligned}$$

where

$$\Psi_{2,byz} = \left(\frac{M\gamma}{2\eta_k} \psi_{byz}^{1/3} + L_2 \psi_\Gamma^{1/3} \right)$$

□

Remark 7.6. *Compared to the non-Byzantine part described in Theorem 7.5, the rate of remains same except for the error floor of the gradient bound suffering a minor terms. It is worth pointing out that under iid data assumption (special case, see Remark 7.2), this error floor is unavoidable, as seen by [135].*

Remark 7.7. *The condition for the step-size η_k remains same as described in the Remark 7.4.*

Remark 7.8. *[Comparison with [135]] In a recent work, [135] provides a perturbed gradient based algorithm to escape the saddle point in non-convex optimization in the presence of Byzantine worker machines. Also, in that paper, the Byzantine resilience is achieved using techniques such as trimmed mean, median and collaborative filtering. These methods require additional assumptions (coordinate of the gradient being sub-exponential etc.) for the purpose of analysis. In this work, we do not require such assumptions. Moreover, we perform a simple norm based thresholding that provides robustness. Also the perturbed gradient descent (PGD) actually requires multiple rounds of communications between the central machine and the worker machines whenever the norm of the gradient is small as this is an indication of either a local minima or a saddle point. In contrast to that, our method does not require any additional communication for escaping the saddle points. Our method provides such ability by virtue of cubic regularization.*

Remark 7.9. *Since our algorithm is second order in nature, it requires less number of iterations compared to the first order gradient based algorithms. Our algorithm achieves a superior rate of $O(1/T^{\frac{2}{3}})$ compared to the gradient based approach of rate $O(1/\sqrt{T})$. Our algorithm dominates ByzantinePGD [135] in terms of convergence, communication rounds and simplicity and efficiency of Byzantine resilience.*

7.5 Experimental Results

First we show that our algorithm indeed *escapes saddle point* with a toy example. We choose a 2 dimensional example: $\min_{\mathbf{w} \in \mathbb{R}^2} [f_1(\mathbf{w}) + f_2(\mathbf{w})]$ where $f_1(\mathbf{w}) = w_1^2 - w_2^2$ and $f_2(\mathbf{w}) = 2w_1^2 - 2w_2^2$ (Here w_i^2 denotes the i -th coordinate of w^2 . This problem is the sum of two non-convex function and has a saddle point at $(0,0)$. In Figure 7.2 (left most) we observe that our algorithm escapes the saddle point $(0,0)$, with random initialization.

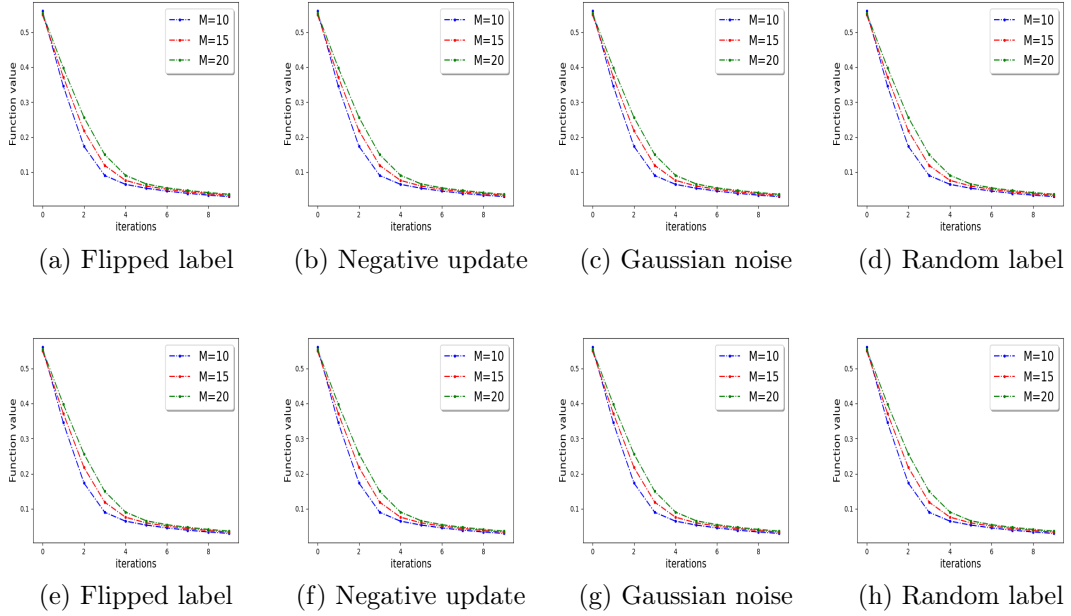


Figure 7.1: Function loss of the training data ‘a9a’ dataset (first row) and ‘w8a’ dataset (second row) with 10%, 15%, 20% Byzantine worker machines for (a,e). Flipped label attack.(b,f). Negative Update attack (c,g). Gaussian noise attack and (d,h). Random label attack for non-convex robust linear regression problem.

Next, we validate our algorithm in Byzantine setup on benchmark LIBSVM ([23]) data-set in both convex and non-convex problems. We choose the following loss functions: (a) Logistic loss: $\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{n} \sum_i \log(1 + \exp(-y_i \mathbf{x}_i^T \mathbf{w})) + \frac{\lambda}{2n} \|\mathbf{w}\|^2$, and (b) Non-convex robust linear regression: $\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{n} \sum_i \log\left(\frac{(y_i - \mathbf{w}^T \mathbf{x}_i)^2}{2} + 1\right)$, where $\mathbf{w} \in \mathbb{R}^d$ is the parameter, $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^d$ are the feature vectors and $\{y_i\}_{i=1}^n \in \{0, 1\}$ are the corresponding labels. We choose ‘a9a’ ($d = 123, n \approx 32K$, we split the data into 70/30

and use as training/testing purpose) and ‘w8a’(training data $d = 300, n \approx 50K$ and testing data $d = 300, n \approx 15K$) classification datasets and partition the data in 20 different worker machines.

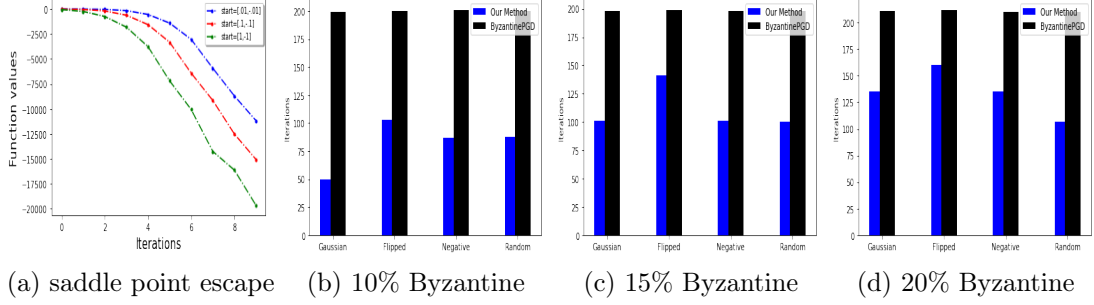


Figure 7.2: (a) Plot of the function value with different initialization to show that the algorithm escapes the saddle point with functional value 0. Comparison of our algorithm with ByzantinePGD [135] in terms of the total number of iterations to achieve small gradient norm ($\|\mathbf{g}\| \leq 0.1$) for (b) 10% (c) 15% and (d) 20% fraction Byzantine machines for different types of attack.

We now show the effectiveness of our algorithm in Byzantine setup. In this work, we consider the following four Byzantine attacks:

1. ‘Gaussian Noise attack’: where the Byzantine worker machines add Gaussian noise to the update.
2. ‘Random label attack’: where the Byzantine worker machines train and learn based on random labels instead of the proper labels.
3. ‘Flipped label attack’: where (for Binary classification) the Byzantine worker machines flip the labels of the data and learn based on wrong labels.
4. ‘Negative update attack’: where the Byzantine workers computes the update \mathbf{s} (here solves the sub-problem in Eq. (7.2)) and communicates $-c * \mathbf{s}$ with $c \in (0, 1)$ making the direction of the update opposite of the actual one.

We show the classification accuracy on testing data of ‘a9a’ and ‘w8a’ dataset for logistic regression problem in Figure 7.3 and training function loss of ‘a9a’ and ‘w8a’

dataset for robust linear regression problem in the Figure 7.1. It is evident from the plots that a simple *norm based thresholding* makes the learning algorithm robust. We choose the parameters $\lambda = 1$, $M = 10$, learning rate $\eta_k = 1$, fraction of the Byzantine machines $\alpha = \{.1, .15, .2\}$ and $\beta = \alpha + \frac{2}{m}$.

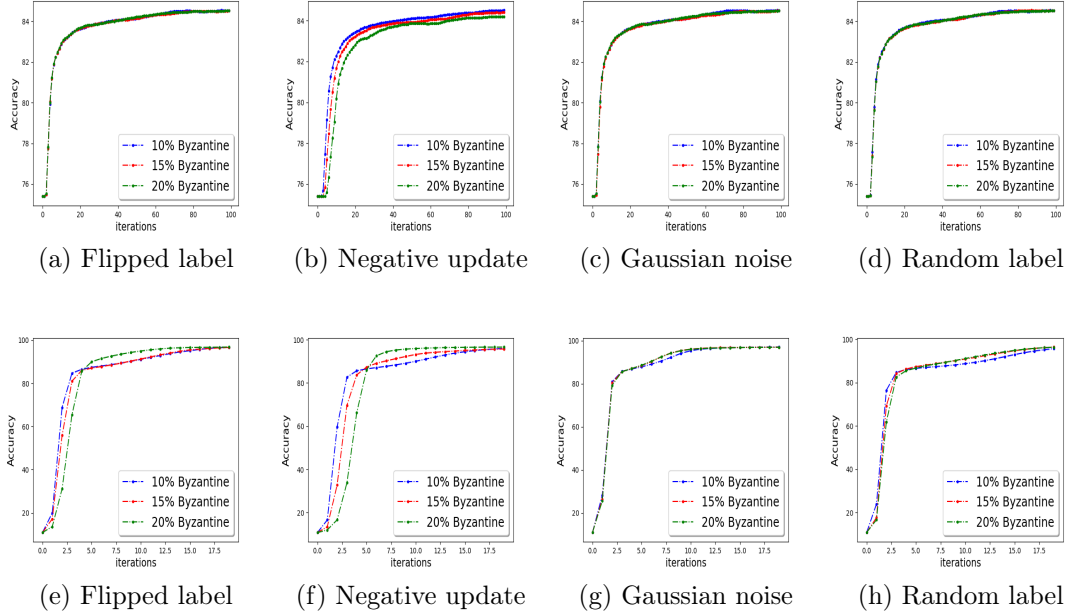


Figure 7.3: Classification accuracy of the testing data ‘a9a’ dataset (first row) and ‘w8a’ dataset (second row) with 10%, 15%, 20% Byzantine worker machines for (a,e). Flipped label.(b,f). Negative Update (c,g). Gaussian noise and (d,h). Random label attack for logistic regression problem.

In Figure 7.4, we show the performance of our algorithm in non-Byzantine setup ($\alpha = \beta = 0$). In the top row of Figure 7.4, we plot the classification accuracy on test data of both ‘a9a’ and ‘w8a’ datasets for logistic regression problem and in the bottom row of Figure 7.4, we plot the function value of the non-convex robust linear regression problem for training data of ‘a9a’ and ‘w8a’ datasets. We choose the learning rate $\eta_k = 1$ and the parameter $\lambda = 1$ and $M = \{10, 15, 20\}$.

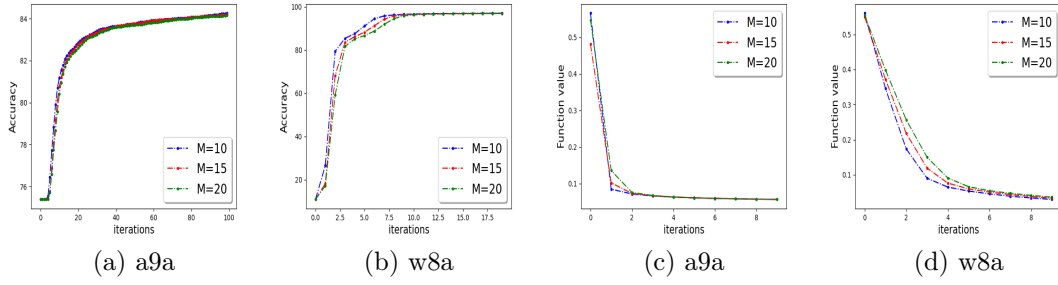


Figure 7.4: (First row) Accuracy of the algorithm for logistic regression on test data of (a). a9a and (b). w8a dataset. (Second row). Function value of the non-convex robust linear regression on the training data of (a). a9a and (b). w8a dataset.

7.6 Conclusion and Future Direction

In this chapter, we address the problem of the saddle points in the non-convex optimization problem in the presence of Byzantine machines. We solve the problem of saddle point escape and 'saddle point attack' in the presence of Byzantine machines with cubic regularized Newton method and norm based thresholding. In each iteration, the worker machines solve a cubic regularized sub-problem that is non-convex in nature. Solving this sub-problem is an interesting challenging problem on its own. In this chapter, we consider that the worker can solve the sub-problem exactly for theoretical analysis purpose. The problem is actually very hard to solve. A few literature works [19, 4] have studied the problem and found efficient solution. In [122], the authors have provided a gradient based approach to solve the sub-problem and showed the analysis for the number of Hessian and gradient computations required to achieve a certain level of convergence guarantee in centralised scenario. The problem is significantly hard for the case of distributed learning as it is very difficult estimate the effect of the update in each machine has on the convergence result. In the future, it would be a challenging to figure out how to solve the problem locally in order to achieve second order stationary problem. Also, a very straight forward extension of this work would be to apply the δ -approximate compressor on the update to reduce the

communication further. Also for the purpose of having a clear idea on the convergence rate, a comparison study between the accelerated method of first order method and cubic newton would be interesting.

BIBLIOGRAPHY

- [1] AWS news blog. <https://tinyurl.com/yxe4hu4w>. Accessed: 2019-10-08.
- [2] Abadi, Martin, Chu, Andy, Goodfellow, Ian, McMahan, H Brendan, Mironov, Ilya, Talwar, Kunal, and Zhang, Li. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016), ACM, pp. 308–318.
- [3] Acharya, Jayadev, De Sa, Chris, Foster, Dylan, and Sridharan, Karthik. Distributed learning with sublinear communication. In *International Conference on Machine Learning* (2019), pp. 40–50.
- [4] Agarwal, Naman, Allen-Zhu, Zeyuan, Bullins, Brian, Hazan, Elad, and Ma, Tengyu. Finding approximate local minima faster than gradient descent. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing* (2017), pp. 1195–1199.
- [5] Agarwal, Naman, Suresh, Ananda Theertha, Yu, Felix Xinnan X, Kumar, Sanjiv, and McMahan, Brendan. cpSGD: Communication-efficient and differentially-private distributed SGD. In *Advances in Neural Information Processing Systems* (2018), pp. 7564–7575.
- [6] Alistarh, Dan, Grubic, Demjan, Li, Jerry, Tomioka, Ryota, and Vojnovic, Milan. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems* (2017), pp. 1709–1720.

- [7] Alistarh, Dan, Grubic, Demjan, Liu, Jerry, Tomioka, Ryota, and Vojnovic, Milan. Communication-efficient stochastic gradient descent, with applications to neural networks.
- [8] Alistarh, Dan, Hoefer, Torsten, Johansson, Mikael, Konstantinov, Nikola, Khirirat, Sarit, and Renggli, Cédric. The convergence of sparsified gradient methods. In *Advances in Neural Information Processing Systems* (2018), pp. 5973–5983.
- [9] Allen-Zhu, Zeyuan, and Li, Yuanzhi. Neon2: Finding local minima via first-order oracles. *arXiv preprint arXiv:1711.06673* (2017).
- [10] Ananthanarayanan, G., Ghodsi, A., Shenker, S., and Stoica, I. Effective straggler mitigation: Attack of the clones. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (NSDI)* (2013), pp. 185–198.
- [11] Bernstein, Jeremy, Wang, Yu-Xiang, Azizzadenesheli, Kamyar, and Anandkumar, Anima. signsgd: Compressed optimisation for non-convex problems. *arXiv preprint arXiv:1802.04434* (2018).
- [12] Bernstein, Jeremy, Wang, Yu-Xiang, Azizzadenesheli, Kamyar, and Anandkumar, Animashree. signSGD: Compressed optimization for non-convex problems. In *International Conference on Machine Learning* (2018), pp. 560–569.
- [13] Bernstein, Jeremy, Zhao, Jiawei, Azizzadenesheli, Kamyar, and Anandkumar, Anima. signsgd with majority vote is communication efficient and byzantine fault tolerant. *arXiv preprint arXiv:1810.05291* (2018).
- [14] Bhojanapalli, Srinadh, Neyshabur, Behnam, and Srebro, Nathan. Global optimality of local search for low rank matrix recovery, 2016.
- [15] Blanchard, Peva, Mhamdi, El Mahdi El, Guerraoui, Rachid, and Stainer, Julien. Byzantine-tolerant machine learning. *arXiv preprint arXiv:1703.02757* (2017).

- [16] Borjesson, P, and Sundberg, C-E. Simple approximations of the error function $q(x)$ for communications applications. *IEEE Transactions on Communications* 27, 3 (1979), 639–643.
- [17] Bubeck, Sébastien. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning* 8 (2017).
- [18] Bubeck, Sébastien, et al. Convex optimization: Algorithms and complexity. *Foundations and Trends[®] in Machine Learning* 8, 3-4 (2015), 231–357.
- [19] Carmon, Yair, and Duchi, John C. Gradient descent efficiently finds the cubic-regularized non-convex newton step. *arXiv preprint arXiv:1612.00547* (2016).
- [20] Cartis, Coralia, Gould, Nicholas IM, and Toint, Philippe L. Adaptive cubic regularisation methods for unconstrained optimization. part i: motivation, convergence and numerical results. *Mathematical Programming* 127, 2 (2011), 245–295.
- [21] Cartis, Coralia, Gould, Nicholas IM, and Toint, Philippe L. Adaptive cubic regularisation methods for unconstrained optimization. part ii: worst-case function-and derivative-evaluation complexity. *Mathematical programming* 130, 2 (2011), 295–319.
- [22] Chang, Chih-Chung, and Lin, Chih-Jen. Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)* 2, 3 (2011), 27.
- [23] Chang, Chih-Chung, and Lin, Chih-Jen. Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)* 2, 3 (2011), 27.

- [24] Charles, Z., Papailiopoulos, D., and Ellenberg, J. Approximate gradient coding via sparse random graphs. *arXiv preprint arXiv:1711.06771* (2017).
- [25] Chen, Wei-Ning, Kairouz, Peter, and Özgür, Ayfer. Breaking the communication-privacy-accuracy trilemma. *CoRR abs/2007.11707* (2020).
- [26] Chen, Xiangyi, Wu, Steven Z, and Hong, Mingyi. Understanding gradient clipping in private sgd: A geometric perspective. *Advances in Neural Information Processing Systems 33* (2020).
- [27] Chen, Yudong, Su, Lili, and Xu, Jiaming. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 1, 2 (2017), 44.
- [28] Chilimbi, Trishul, Suzue, Yutaka, Apacible, Johnson, and Kalyanaraman, Karthik. Project adam: Building an efficient and scalable deep learning training system. In *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)* (2014), pp. 571–582.
- [29] Choromanska, Anna, Henaff, Mikael, Mathieu, Michael, Arous, G  rard Ben, and LeCun, Yann. The loss surfaces of multilayer networks, 2015.
- [30] Cohen, G  rard, Honkala, Iiro, Litsyn, Simon, and Lobstein, Antoine. *Covering codes*, vol. 54. Access Online via Elsevier, 1997.
- [31] Costa, Paolo, Ballani, Hitesh, Razavi, Kaveh, and Kash, Ian. R2c2: A network stack for rack-scale computers. In *SIGCOMM 2015* (August 2015), ACM - Association for Computing Machinery.
- [32] Crane, Rixon, and Roosta, Fred. Dingo: Distributed Newton-type method for gradient-norm optimization. In *Advances in Neural Information Processing Systems* (2019).

- [33] Damaskinos, Georgios, El Mhamdi, El Mahdi, Guerraoui, Rachid, Guirguis, Arsany Hany Abdelmessih, and Rouault, SÃ©bastien Louis Alexandre. Aggregathor: Byzantine machine learning via robust gradient aggregation. 19. Published in the Conference on Systems and Machine Learning (SysML) 2019, Stanford, CA, USA.
- [34] Dauphin, Yann N, Pascanu, Razvan, Gulcehre, Caglar, Cho, Kyunghyun, Ganguli, Surya, and Bengio, Yoshua. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems* (2014), vol. 27, pp. 2933–2941.
- [35] Dean, J., and Barroso, L. A. The tail at scale. *Communications of the ACM* 56, 2 (2013), 74–80.
- [36] Dean, J., and Ghemawat, S. MapReduce: Simplified data processing on large clusters. *Communications of the ACM* 51, 1 (Jan. 2008), 107–113.
- [37] Derezhinski, Michal, and Mahoney, Michael W. Distributed estimation of the inverse hessian by determinantal averaging. In *Advances in Neural Information Processing Systems* (2019), pp. 11401–11411.
- [38] Drineas, Petros, Magdon-Ismail, Malik, Mahoney, Michael W, and Woodruff, David P. Fast approximation of matrix coherence and statistical leverage. *Journal of Machine Learning Research* 13, Dec (2012), 3475–3506.
- [39] Drineas, Petros, and Mahoney, Michael W. Randnla: randomized numerical linear algebra. *Communications of the ACM* 59, 6 (2016), 80–90.
- [40] Du, Simon S, Jin, Chi, Lee, Jason D, Jordan, Michael I, Póczos, Barnabas, and Singh, Aarti. Gradient descent can take exponential time to escape saddle points. *arXiv preprint arXiv:1705.10412* (2017).

- [41] Dutta, S., Cadambe, V., and Grover, P. Short-dot: Computing large linear transforms distributedly using coded short dot products. In *Advances in Neural Information Processing Systems* (2016), pp. 2100–2108.
- [42] Dutta, S., Cadambe, V., and Grover, P. Coded convolution for parallel and distributed computing within a deadline. *arXiv preprint arXiv:1705.03875* (2017).
- [43] Dwork, Cynthia, McSherry, Frank, Nissim, Kobbi, and Smith, Adam. Calibrating noise to sensitivity in private data analysis. *Journal of Privacy and Confidentiality* 7, 3 (2016), 17–51.
- [44] Dwork, Cynthia, Roth, Aaron, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.
- [45] Erlingsson, Úlfar, Pihur, Vasyi, and Korolova, Aleksandra. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security* (2014), ACM, pp. 1054–1067.
- [46] Fallah, Alireza, Mokhtari, Aryan, and Ozdaglar, Asuman. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948* (2020).
- [47] Figueiredo, M., Nowak, R. D., and Wright, S. J. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of selected topics in signal processing* 1, 4 (2007), 586–597.
- [48] Gallager, R. Low-density parity-check codes. *IRE Transactions on information theory* 8, 1 (1962), 21–28.

- [49] Gandikota, Venkata, Maity, Raj Kumar, and Mazumdar, Arya. vqsgd: Vector quantized stochastic gradient descent. *arXiv preprint arXiv:1911.07971* (2019).
- [50] Garg, R., and Khandekar, R. Gradient descent with sparsification: an iterative algorithm for sparse recovery with restricted isometry property. In *Proceedings of the 26th Annual International Conference on Machine Learning* (2009), ACM, pp. 337–344.
- [51] Ge, Rong, Jin, Chi, and Zheng, Yi. No spurious local minima in nonconvex low rank problems: A unified geometric analysis. In *Proceedings of the 34th International Conference on Machine Learning* (International Convention Centre, Sydney, Australia, 06–11 Aug 2017), vol. 70 of *Proceedings of Machine Learning Research*, PMLR, pp. 1233–1242.
- [52] Ghosh, Avishek, Hong, Justin, Yin, Dong, and Ramchandran, Kannan. Robust federated learning in a heterogeneous environment. *arXiv preprint arXiv:1906.06629* (2019).
- [53] Ghosh, Avishek, Maity, Raj Kumar, Kadhe, Swanand, Mazumdar, Arya, and Ramchandran, Kannan. Communication-efficient and byzantine-robust distributed learning. *arXiv preprint arXiv:1911.09721* (2019).
- [54] Ghosh, Avishek, Maity, Raj Kumar, Kadhe, Swanand, Mazumdar, Arya, and Ramchandran, Kannan. Communication-efficient and byzantine-robust distributed learning. *arXiv preprint arXiv:1911.09721* (2019).
- [55] Ghosh, Avishek, Maity, Raj Kumar, Kadhe, Swanand, Mazumdar, Arya, and Ramchandran, Kannan. Communication-efficient and byzantine-robust distributed learning. In *2020 Information Theory and Applications Workshop (ITA)* (2020), IEEE, pp. 1–28.

- [56] Ghosh, Avishek, Maity, Raj Kumar, and Mazumdar, Arya. Distributed newton can communicate less and resist byzantine workers. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual* (2020).
- [57] Gupta, Vipul, Kadhe, Swanand, Courtade, Thomas, Mahoney, Michael W, and Ramchandran, Kannan. Oversketched newton: Fast convex optimization for serverless systems. *arXiv preprint arXiv:1903.08857* (2019).
- [58] Halbawi, W., Ruhi, N. Azizan, Salehi, F., and Hassibi, B. Improving distributed gradient descent using Reed-Solomon codes. *arXiv preprint arXiv:1706.05436* (2017).
- [59] Hardt, Moritz, and Recht, Benjamin. Patterns, predictions, and actions: A story about machine learning. *arXiv preprint arXiv:2102.05242* (2021).
- [60] Haykin, Simon. *An introduction to analog and digital communication*. John Wiley, 1994.
- [61] Horváth, Samuel, Kovalev, Dmitry, Mishchenko, Konstantin, Stich, Sebastian, and Richtárik, Peter. Stochastic Distributed Learning with Gradient Quantization and Variance Reduction. *arXiv e-prints* (Apr 2019), arXiv:1904.05115.
- [62] Ivkin, Nikita, Rothchild, Daniel, Ullah, Enayat, Stoica, Ion, Arora, Raman, et al. Communication-efficient distributed SGD with sketching. In *Advances in Neural Information Processing Systems* (2019), pp. 13144–13154.
- [63] Jain, Prateek, Jin, Chi, Kakade, Sham M., and Netrapalli, Praneeth. Global convergence of non-convex gradient descent for computing matrix squareroot, 2017.

- [64] Jin, Chi, Ge, Rong, Netrapalli, Praneeth, Kakade, Sham M, and Jordan, Michael I. How to escape saddle points efficiently. In *International Conference on Machine Learning* (2017), PMLR, pp. 1724–1732.
- [65] Kakade, S. M., Kanade, V., Shamir, O., and Kalai, A. Efficient learning of generalized linear and single index models with isotonic regression. In *Advances in Neural Information Processing Systems* (2011), pp. 927–935.
- [66] Kalan, Seyed Mohammadreza Mousavi, Soltanolkotabi, Mahdi, and Avestimehr, A Salman. Fitting ReLus via SGD and quantized SGD. In *2019 IEEE International Symposium on Information Theory (ISIT)* (2019), IEEE, pp. 2469–2473.
- [67] Karakus, C., Sun, Y., Diggavi, S., and Yin, W. Straggler mitigation in distributed optimization through data encoding. In *Advances in Neural Information Processing Systems* (2017), pp. 5440–5448.
- [68] Karimireddy, Sai Praneeth, Kale, Satyen, Mohri, Mehryar, Reddi, Sashank, Stich, Sebastian, and Suresh, Ananda Theertha. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning* (2020), PMLR, pp. 5132–5143.
- [69] Karimireddy, Sai Praneeth, Kale, Satyen, Mohri, Mehryar, Reddi, Sashank J, Stich, Sebastian U, and Suresh, Ananda Theertha. Scaffold: Stochastic controlled averaging for on-device federated learning. *arXiv preprint arXiv:1910.06378* (2019).
- [70] Karimireddy, Sai Praneeth, Rebjock, Quentin, Stich, Sebastian, and Jaggi, Martin. Error feedback fixes signSGD and other gradient compression schemes. In *International Conference on Machine Learning* (2019), pp. 3252–3261.

- [71] Karimireddy, Sai Praneeth, Rebjock, Quentin, Stich, Sebastian U, and Jaggi, Martin. Error feedback fixes signsgd and other gradient compression schemes. *arXiv preprint arXiv:1901.09847* (2019).
- [72] Kawaguchi, Kenji. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems* (2016), D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29, Curran Associates, Inc., pp. 586–594.
- [73] Kohler, Jonas Moritz, and Lucchi, Aurelien. Sub-sampled cubic regularization for non-convex optimization. In *International Conference on Machine Learning* (2017), PMLR, pp. 1895–1904.
- [74] Koloskova, Anastasiia, Stich, Sebastian Urban, and Jaggi, Martin. Decentralized stochastic optimization and gossip algorithms with compressed communication. *Proceedings of Machine Learning Research* 97, CONF (2019).
- [75] Koltchinskii, V., Lounici, K., and Tsybakov, A. B. Nuclear-norm penalization and optimal rates for noisy low-rank matrix completion. *The Annals of Statistics* 39, 5 (2011), 2302–2329.
- [76] Konečný, Jakub, McMahan, H Brendan, Ramage, Daniel, and Richtárik, Peter. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527* (2016).
- [77] Konečný, Jakub, McMahan, H Brendan, Yu, Felix X, Richtárik, Peter, Suresh, Ananda Theertha, and Bacon, Dave. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).
- [78] Krizhevsky, Alex. Learning multiple layers of features from tiny images.
- [79] Lamport, Leslie, Shostak, Robert, and Pease, Marshall. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.* 4, 3 (July 1982), 382–401.

- [80] LeCun, Yann, Bottou, Léon, Bengio, Yoshua, Haffner, Patrick, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86, 11 (1998), 2278–2324.
- [81] LeCun, Yann, and Cortes, Corinna. MNIST handwritten digit database.
- [82] Lee, Jason D, Panageas, Ioannis, Piliouras, Georgios, Simchowitz, Max, Jordan, Michael I, and Recht, Benjamin. First-order methods almost always avoid saddle points. *arXiv preprint arXiv:1710.07406* (2017).
- [83] Lee, Jason D, Simchowitz, Max, Jordan, Michael I, and Recht, Benjamin. Gradient descent converges to minimizers. *arXiv preprint arXiv:1602.04915* (2016).
- [84] Lee, K., Lam, M., Pedarsani, R., Papailiopoulos, D., and Ramchandran, K. Speeding up distributed machine learning using codes. *IEEE Transactions on Information Theory* 64, 3 (March 2018), 1514–1529.
- [85] Lee, K., Pedarsani, R., Papailiopoulos, D., and Ramchandran, K. Coded computation for multicore setups. In *2017 IEEE International Symposium on Information Theory (ISIT)* (June 2017), pp. 2413–2417.
- [86] Lee, K., Suh, C., and Ramchandran, K. High-dimensional coded matrix multiplication. In *Proceedings of IEEE International Symposium on Information Theory (ISIT)* (2017), pp. 2418–2422.
- [87] MacWilliams, Florence Jessie, and Sloane, Neil James Alexander. *The theory of error-correcting codes*, vol. 16. Elsevier, 1977.
- [88] Mahoney, Michael W. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning* 3, 2 (2011), 123–224.

- [89] Mairal, J., Bach, F., Ponce, J., and Sapiro, G. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning* (2009), ACM, pp. 689–696.
- [90] Mayekar, Prathamesh, and Tyagi, Himanshu. Ratq: A universal fixed-length quantizer for stochastic optimization. *arXiv preprint arXiv:1908.08200* (2019).
- [91] Mayekar, Prathamesh, and Tyagi, Himanshu. Limits on gradient compression for stochastic optimization. *arXiv preprint arXiv:2001.09032* (2020).
- [92] Mhamdi, El Mahdi El, Guerraoui, Rachid, and Rouault, Sébastien. The hidden vulnerability of distributed learning in byzantium. *arXiv preprint arXiv:1802.07927* (2018).
- [93] Nemirovski, A., Juditsky, A., Lan, G., and Shapiro, A. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization* 19, 4 (2009), 1574–1609.
- [94] Nesterov, Yurii, and Polyak, Boris T. Cubic regularization of newton method and its global performance. *Mathematical Programming* 108, 1 (2006), 177–205.
- [95] Pilanci, Mert, and Wainwright, Martin J. Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. *SIAM Journal on Optimization* 27, 1 (2017), 205–245.
- [96] Plan, Y., and Vershynin, R. The generalized lasso with non-linear observations. *IEEE Transactions on information theory* 62, 3 (2016), 1528–1537.
- [97] Qian, Jiang, Wu, Yuren, Zhuang, Bojin, Wang, Shaojun, Xiao, Jing, et al. Understanding gradient clipping in incremental gradient methods. In *International Conference on Artificial Intelligence and Statistics* (2021), PMLR, pp. 1504–1512.

- [98] Raviv, N., Tamo, I., Tandon, R., and Dimakis, A. G. Gradient coding from cyclic MDS codes and expander graphs. *arXiv preprint arXiv:1707.03858* (2017).
- [99] Reddi, Sashank J, Konečný, Jakub, Richtárik, Peter, Póczós, Barnabás, and Smola, Alex. Aide: Fast and communication efficient distributed optimization. *arXiv preprint arXiv:1608.06879* (2016).
- [100] Richardson, T., and Urbanke, R. L. *Modern Coding Theory*. Cambridge University Press, New York, NY, USA, 2008.
- [101] Richardson, T. J., and Urbanke, R. L. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Transactions on Information Theory* 47, 2 (Feb 2001), 599–618.
- [102] Roosta, Fred, Liu, Yang, Xu, Peng, and Mahoney, Michael W. Newton-mr: Newton’s method without smoothness or convexity. *arXiv preprint arXiv:1810.00303* (2018).
- [103] Seide, Frank, Fu, Hao, Droppo, Jasha, Li, Gang, and Yu, Dong. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Fifteenth Annual Conference of the International Speech Communication Association* (2014).
- [104] Shah, N. B., Lee, K., and Ramchandran, K. When do redundant requests reduce latency? *IEEE Transactions on Communications* 64, 2 (Feb 2016), 715–722.
- [105] Shalev-Shwartz, S., and Ben-David, S. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, New York, NY, USA, 2014.
- [106] Shalev-Shwartz, Shai, and Ben-David, Shai. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

- [107] Shalev-Shwartz, Shai, Srebro, Nathan, and Zhang, Tong. Trading accuracy for sparsity in optimization problems with sparsity constraints. *SIAM Journal on Optimization* 20, 6 (2010), 2807–2832.
- [108] Shamir, Ohad, Srebro, Nati, and Zhang, Tong. Communication-efficient distributed optimization using an approximate newton-type method. In *International conference on machine learning* (2014), pp. 1000–1008.
- [109] Shamir, Ohad, Srebro, Nati, and Zhang, Tong. Communication-efficient distributed optimization using an approximate newton-type method. In *International conference on machine learning* (2014), pp. 1000–1008.
- [110] Shokri, Reza, and Shmatikov, Vitaly. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security* (2015), ACM, pp. 1310–1321.
- [111] Sipser, M., and Spielman, D. A. Expander codes. *IEEE Transactions on Information Theory* 42, 6 (Nov 1996), 1710–1722.
- [112] Soudry, Daniel, and Carmon, Yair. No bad local minima: Data independent training error guarantees for multilayer neural networks, 2016.
- [113] Stich, Sebastian U, Cordonnier, Jean-Baptiste, and Jaggi, Martin. Sparsified SGD with memory. In *Advances in Neural Information Processing Systems* (2018), pp. 4447–4458.
- [114] Strom, Nikko. Scalable distributed DNN training using commodity GPU cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association* (2015).

- [115] Su, Lili, and Vaidya, Nitin H. Fault-tolerant multi-agent optimization: optimal iterative distributed algorithms. In *Proceedings of the 2016 ACM symposium on principles of distributed computing* (2016), ACM, pp. 425–434.
- [116] Sun, Ju, Qu, Qing, and Wright, John. A geometric analysis of phase retrieval. *CoRR abs/1602.06664* (2016).
- [117] Sun, Ju, Qu, Qing, and Wright, John. Complete dictionary recovery over the sphere i: Overview and the geometric picture. *IEEE Transactions on Information Theory* 63, 2 (Feb 2017), 853–884.
- [118] Suresh, Ananda Theertha, Yu, Felix X, Kumar, Sanjiv, and McMahan, H Brendan. Distributed mean estimation with limited communication. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), JMLR. org, pp. 3329–3337.
- [119] Swarm2. Swarm user documentation. <https://people.cs.umass.edu/~swarm/index.php?n=Main.NewSwarmDoc>, 2018. Accessed: 2018-01-05.
- [120] Tandon, R., Lei, Q., Dimakis, A. G., and Karampatziakis, N. Gradient coding: Avoiding stragglers in distributed learning. In *Proceedings of the 34th International Conference on International Conference on Machine Learning (ICML)* (2017), pp. 3368–3376.
- [121] Tibshirani, R., Wainwright, M., and Hastie, T. *Statistical learning with sparsity: the lasso and generalizations*. Chapman and Hall/CRC, 2015.
- [122] Tripuraneni, N, Stern, M, Jin, C, Regier, J, and Jordan, MI. Stochastic cubic regularization for fast nonconvex optimization. In *Advances in Neural Information Processing Systems* (2018), pp. 2899–2908.

- [123] Wainwright, Martin J. *High-dimensional statistics: A non-asymptotic viewpoint*, vol. 48. Cambridge University Press, 2019.
- [124] Wang, D., Joshi, G., and Wornell, G. Using straggler replication to reduce latency in large-scale parallel computing. *ACM SIGMETRICS Performance Evaluation Review* 43, 3 (2015), 7–11.
- [125] Wang, Hongyi, Sievert, Scott, Liu, Shengchao, Charles, Zachary, Papailiopoulos, Dimitris, and Wright, Stephen. Atomo: Communication-efficient learning via atomic sparsification. In *Advances in Neural Information Processing Systems* (2018), pp. 9850–9861.
- [126] Wang, Shusen, Gittens, Alex, and Mahoney, Michael W. Sketched ridge regression: Optimization perspective, statistical perspective, and model averaging. *The Journal of Machine Learning Research* 18, 1 (2017), 8039–8088.
- [127] Wang, Shusen, Roosta-Khorasani, Farbod, Xu, Peng, and Mahoney, Michael W. Giant: Globally improved approximate newton method for distributed optimization. In *Advances in Neural Information Processing Systems* (2018), pp. 2332–2342.
- [128] Wang, Zhe, Zhou, Yi, Liang, Yingbin, and Lan, Guanghai. Stochastic variance-reduced cubic regularization for nonconvex optimization. In *The 22nd International Conference on Artificial Intelligence and Statistics* (2019), PMLR, pp. 2731–2740.
- [129] Wang, Zhe, Zhou, Yi, Liang, Yingbin, and Lan, Guanghai. Cubic regularization with momentum for nonconvex optimization. In *Uncertainty in Artificial Intelligence* (2020), PMLR, pp. 313–322.

- [130] Warner, Stanley L. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association* 60, 309 (1965), 63–69.
- [131] Wen, Wei, Xu, Cong, Yan, Feng, Wu, Chunpeng, Wang, Yandan, Chen, Yiran, and Li, Hai. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in neural information processing systems* (2017), pp. 1509–1519.
- [132] Wu, Yihong. Lecture notes for ece598yw: Information-theoretic methods for high-dimensional statistics, 2017.
- [133] Xu, Yi, Jin, Rong, and Yang, Tianbao. First-order stochastic algorithms for escaping from saddle points in almost linear time. *arXiv preprint arXiv:1711.01944* (2017).
- [134] Yang, Y., Grover, P., and Kar, S. Coding method for parallel iterative linear solver. *arXiv preprint arXiv:1706.00163* (2017).
- [135] Yin, Dong, Chen, Yudong, Kannan, Ramchandran, and Bartlett, Peter. Byzantine-robust distributed learning: Towards optimal statistical rates. In *Proceedings of the 35th International Conference on Machine Learning* (Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018), Jennifer Dy and Andreas Krause, Eds., vol. 80 of *Proceedings of Machine Learning Research*, PMLR, pp. 5650–5659.

- [136] Yin, Dong, Chen, Yudong, Kannan, Ramchandran, and Bartlett, Peter. Defending against saddle point attack in Byzantine-robust distributed learning. In *Proceedings of the 36th International Conference on Machine Learning* (Long Beach, California, USA, 09–15 Jun 2019), Kamalika Chaudhuri and Ruslan Salakhutdinov, Eds., vol. 97 of *Proceedings of Machine Learning Research*, PMLR, pp. 7074–7084.
- [137] Yu, Q., Maddah-Ali, M. A., and Avestimehr, A. S. Polynomial codes: an optimal design for high-dimensional coded matrix multiplication. *arXiv preprint arXiv:1705.10464* (2017).
- [138] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. Spark: Cluster computing with working sets. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing (HotCloud)* (2010), pp. 10–10.
- [139] Zhang, Yuchen, Duchi, John, Jordan, Michael I, and Wainwright, Martin J. Information-theoretic lower bounds for distributed statistical estimation with communication constraints. In *Advances in Neural Information Processing Systems* (2013), pp. 2328–2336.
- [140] Zhang, Yuchen, and Lin, Xiao. Disco: Distributed optimization for self-concordant empirical loss. In *International conference on machine learning* (2015), pp. 362–370.
- [141] Zhou, Dongruo, Xu, Pan, and Gu, Quanquan. Stochastic variance-reduced cubic regularized newton methods. In *International Conference on Machine Learning* (2018), PMLR, pp. 5990–5999.